Net-Centric Implementation Framework

Part 1: Overview

Part 2: Traceability

Part 3: Migration Guidance

Part 4: Node Guidance

Part 5: Developer Guidance

Part 6: Contracting Guidance for Acquisition

V 2.2.0

17 June 2008

Net-Centric Enterprise Solutions for Interoperability (NESI) is a collaborative activity of the USN Program Executive Office for Command, Control, Communications, Computers and Intelligence (PEO C4I); the USAF Electronic Systems Center (ESC); and the Defense Information Systems Agency (DISA).

NESI-X Subdocument generated using: **View, NESI Part 2: Traceability**
Generated: **Mon Jun 23 11:54:34 PDT 2008**
NESI-X Version: **v1.1.9 build 3309 - 2008/05/16 08:17**

Table of Contents

Perspectives

P1117: NESI Executive Summary

Net-Centric Enterprise Solutions for Interoperability (***NESI***) provides, for all phases of the acquisition of net-centric solutions, actionable guidance that meets DoD Network-Centric Warfare goals. The guidance in NESI is derived from the higher level, more abstract concepts provided in various directives, policies and mandates such as the Net-Centric Operations and Warfare Reference Model (NCOW RM) [R1176] and the ASD(NII) Net-Centric Checklist [R1177] . As currently structured, NESI implementation covers architecture, design and implementation; compliance checklists; and a collaboration environment that includes a repository.

More specifically, NESI is a body of architectural and engineering knowledge that guides the design, implementation, maintenance, evolution, and use of the Information Technology (IT) portion of net-centric solutions for military application. NESI provides specific technical recommendations that a DoD organization can use as references. Stated another way, NESI serves as a reference set of compliant instantiations of these directives.

NESI is derived from a studied examination of enterprise-level needs and, more importantly, from the collective practical experience of recent and on-going program-level implementations. It is based on today's technologies and probable near-term technology developments. It describes the practical experience of system developers within the context of a minimal top-down technical framework. Most, if not all, of the guidance in NESI is in line with commercial best practices in the area of enterprise computing.

NESI applies to all phases of the acquisition process as defined in DoD Directive 5000.1 [R1164] and DoD Instruction 5000.2 [R1165] and to both new and legacy programs. NESI provides explicit counsel for building in net-centricity from the ground up and for migrating legacy systems to greater degrees of net-centricity.

NESI subsumes a number of references and directives; in particular, the Air Force C2 Enterprise Technical Reference Architecture (C2ERA) and the Navy Reusable Applications Integration and Development Standards (RAPIDS). Initial authority for NESI is per the Memorandum of Agreement between Commander, Space and Naval Warfare Systems Command (SPAWAR); Navy Program Executive Officer, C4I & Space (now PEO C4I); and the United States Air Force Electronic Systems Center (ESC), dated 22 December 2003, Subject: Cooperation Agreement for Net-Centric Solutions for Interoperability (NESI). The Defense Information Systems Agency (DISA) formally joined the NESI effort in 2006.

Content Structure

Perspectives	NESI ***Perspectives*** describe a topic and encompass related, more specific Perspectives or encapsulate a set of Guidance and Best Practice details, Examples, References, and Glossary entries that pertain to the topic.
Guidance	NESI ***Guidance*** is in the form of atomic, succinct, absolute and definitive Statements related to one or more Perspectives. Each Guidance Statement is linked to Guidance Details which provide Rationale, relationships with other Guidance or Best Practices, and Evaluation Criteria with one or more Tests, Procedures and Examples which facilitate validation of using the Guidance through observation, measurement or other means. Guidance Statements are intended to be binding in nature, especially if used as part of a Statement of Work (SOW) or performance specification.
Best Practices	NESI ***Best Practices*** are advisory in nature to assist program or project managers and personnel. Best Practice Details can have all the same parts as NESI Guidance. The use of

	NESI Best Practices are at the discretion of the program or project manager.
Examples	NESI ***Examples*** illustrate key aspects of Perspectives, Guidance, or Best Practices.
Glossary	NESI ***Glossary*** entries provide terms, acronyms, and definitions used in The context of NESI Perspectives, Guidance and Best Practices.
References	NESI ***References*** identify directives, instructions, books, Web sites, and other sources of information useful for planning or execution.

Releasability Statement

NESI ***Net-Centric Implementation*** v2.2 has been cleared for public release by competent authority in accordance with DoD Directive 5230.9 [R1232] and is granted Distribution Statement A: Approved for public release; distribution is unlimited. Obtain electronic copies of this document at http://nesipublic.spawar.navy.mil.

Vendor Neutrality

The NESI documentation sometimes refers to specific vendors and their products in the context of examples and lists. However, NESI is vendor-neutral. Mentioning a vendor or product is not intended as an endorsement, nor is a lack of mention intended as a lack of endorsement. Code examples typically use open-source products since NESI is built on the open-source philosophy. NESI accepts inputs from multiple sources so the examples tend to reflect whatever tools the contributor was using or knew best. However, the products described are not necessarily the best choice for every circumstance. Users are encouraged to analyze specific project requirements and choose tools accordingly. There is no need to obtain, or ask contractors to obtain, the tools that appear as examples in this guide. Similarly, any lists of products or vendors are intended only as references or starting points, and not as a list of recommended or mandated options.

Disclaimer

Every effort has been made to make NESI documentation as complete and accurate as possible. Even with frequent updates, this documentation may not always immediately reflect the latest technology or guidance. Also, references and links to external material are as accurate as possible; however, they are subject to change or may have additional access requirements such as Public Key Infrastructure (PKI) certificates, Common Access Card (CAC) for user identification, and user account registration.

Contributions and Comments

NESI is an open project that involves the entire development community. Anyone is welcome to contribute comments, corrections, or relevant knowledge to the guides via the Change Request tab on the NESI Public site, http://nesipublic.spawar.navy.mil, or via the following email address: nesi@spawar.navy.mil.

P1288: Part 2: Traceability

Part 2: Traceability provides a mapping of specific NESI Guidance to other, often more general, high-level DoD net-centric and interoperability efforts such as the Assistant Secretary of Defense for Networks and Information Integration/ Department of Defense Chief Information Officer, or ASD(NII)/DoD CIO, Net-Centric Checklist. [R1177] Part 2 includes Perspectives that follow the structure of each high-level effort and provide a NESI interpretation of the implementation implications for program managers and developers which these other efforts direct or imply. These Perspectives, and the associated NESI Guidance and Best Practice links, provide a means of navigating NESI content based on the traceability Part 2 provides. The efforts to which Part 2 content traces may be DoD- or Service-specific; Part 2 currently traces to the following.

Detailed Perspectives

P1239: ASD(NII): Net-Centric Guidance

The ***ASD(NII) Checklist Guidance*** is primarily for managers of new programs or programs that are undergoing a transformation or major upgrade and is especially useful in the pre-systems acquisition and systems acquisition phases. The ASD(NII) Net-Centric Checklist [R1177] uses net-centric design precepts called ***tenets*** to guide the move into the net-centric environment. The design tenets help the DoD leadership understand how net-centricity is evolving. NESI provides specific technical direction for satisfying the Net-Centric Checklist. Note that some tenets address doctrinal or procedural requirements; NESI guidance does not address those areas.

Intended Audience

The Net-Centric Guidance is primarily applicable for new programs or programs that are undergoing a transformation or major upgrade, especially in the pre-systems acquisition and systems acquisition phases. The intended audience for this document includes the following:

- Program managers
- Deputy program managers
- Contracting officers
- Chief engineers
- Contractor personnel
- Enterprise and software architects

Detailed Perspectives

The following perspectives address the ASD(NII) Net-Centric Checklist design tenet categories.

Data

Services

Information Assurance/Security

Transport

Each design tenet provides specific technical guidance to enable the system to satisfy its net-centric requirements.

The technical guidance in Part 2 is not necessarily all encompassing; rather, use these guidance statements as part of the overall system engineering analysis of a program to facilitate the evolution of a program or project to net-centricity. Additionally, not all design tenets can be satisfied strictly by technical guidance. All elements of **Doctrine, Organization, Training, Materiel, Leadership, Personnel, and Facilities** (**DOTMLPF**) must participate in the evolution of net-centricity.

P1244: Data

The DoD Net-Centric Data Strategy [R1172] is a key enabler of DoD transformation. Significant attributes of the data strategy include the following:

- Ensuring that data are understandable and trustable, and that they are visible and accessible when and where needed to accelerate decision-making.
- "Tagging" data (intelligence, non-intelligence, raw, and processed) with metadata that supports discovery by both known and unanticipated users in the enterprise.
- Posting data to shared spaces that all users can access, except when limited by security, policy, or regulations.
- Posting in parallel with processing; Task/Post/Process/Use replaces the Task/Process/Exploit/Disseminate paradigm.
- Separating data from applications so that users may choose different applications to exploit the same data.
- Handling information only once to eliminate duplicate, non-authoritative data.

Note: *This section explains the design tenets surrounding data and data assets. A data asset is any entity that involves data. For example, a database is a data asset composed of data records.*

Detailed Perspectives

P1250: Design Tenet: Make Data Visible

Data visibility requires an integrated environment of metadata models about the data assets. A data asset is visible when discovery metadata that describes the asset is accessible. Perform forward and/or reverse engineering to capture metadata that describes the data assets of a **node**. Making data visible (even if not accessible) helps develop information about the node and its applications through insights such as the following:

- Essential missions that define the reason for the enterprise; the ultimate goals and objectives that measure enterprise accomplishment
- Procedures performed by various groups in the enterprise that achieve these essential missions
- The specific databases, information systems, and processes that groups use to accomplish aspects of the essential missions
- Context-independent semantic templates of data elements and mechanisms for configuring into data models, as determined by subject matter experts
- Mechanisms for configuring data models into databases used by organizations in the enterprise

Considerations

- Make all data assets visible, even if they are not accessible.
- Use the DoD Discovery Metadata Specification (DDMS) [R1225] and all of its attributes to describe data assets.
- If possible, generate discovery metadata automatically.

Guidance

- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1125: Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1863: Make shareable data assets visible, even if they are not accessible.
- BP1865: Provide sufficient program, project, or initiative **metadata** descriptions and automated support to enable **mediation** and translation of the data between **interfaces**.

P1252: Design Tenet: Make Data Accessible

Data accessibility requires defining data assets that exist within acceptable boundaries of security, along with the information necessary to access them. **Relational databases** automatically contain metadata about data assets. This perspective extends that definition to **XML** data that may exist independently or that are mapped to and/or from relational data. The following considerations focus on using XML; however, there are alternatives (see the final two Considerations).

Considerations

XML Requirement

- Use XML to exchange information across systems. Define and implement an XML version of each external interface in all systems. If a system makes data available to external partners, make that data available in the form of an XML document. This is required even if none of the current known partners want or send XML data. Systems may implement other external data exchange mechanisms if an XML interface is supported. Systems may implement other external data exchange mechanisms in addition to an XML interface.

XML Interface Specification

- The system that defines an XML interface will do the following:
    - Specify the syntax of the XML documents it accepts and produces
    - Use the XML Schema standard to express these specifications. Refer to XML Schema Best Practices [R1226] for guidance on creating XML schemas
    - Enter the schema in the DoD Metadata Registry and Clearinghouse. [R1227] This should occur as early as possible in the development process. Consult designated DoD XML Namespace Managers for guidance in choosing element, attribute, and type identifiers
- An XML interface is responsible for the following actions:
    - Accept input data, producing output data, or both
    - Encode this data in XML documents
    - Specify the schema of the XML documents it accepts and produces
    - Provide documentation that allows programmers and users to understand the meaning of those documents
    - Be implemented by a runtime service that accepts and produces such documents

XML Interface Useage

- A system that uses an XML interface defined by some other system shall record this fact in the DoD Metadata Registry and Clearinghouse.

XML Transport

- Systems must implement one version of each XML interface that is accessible through a URL using HTTP/HTTPS. Systems may implement other versions of the interface using other transport mechanisms, such as **FTP** or **SMTP**, as long as they also support the HTTP version.

Open-Standard Alternatives to XML Format

- Information that is customarily exchanged using a well-known open-standard format does not have to be made available in XML. For example, systems may transfer image data in JPEG format, and email messages may continue to use RFC 822 (***Standards for ARPA Internet Text Messages***) headers. It is not necessary to develop an equivalent XML interface for these. Make a list of the exception formats available. It is not necessary to convert information intended for presentation that is currently held in Standard Generalized Markup Language (SGML) format immediately into XML. However, systems should consider future migration from SGML to XML.

Proprietary Alternatives to XML Format

- Information that can only be expressed using closed proprietary formats does not have to be made available in XML. For example, systems may continue to exchange word processor files in Microsoft® Word (DOC format); it is not necessary to develop an equivalent XML interface for this information.

Guidance

- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1763: Indicate the security classification for all classified data.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1390: Standardize on the terminology published by relevant **Communities of Interest** (**COIs**) listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1874: Develop methods to forward IP datagrams from external networks.

P1253: Design Tenet: Make Data Understandable

Use well-defined standard data elements to establish the semantic basis for data models. To enable data understanding, start with well-defined data ontologies, taxonomies, and vocabularies using standard data elements as the basis for data model structure templates used throughout database models and operating databases. The use of standard data elements also extends to the semantics of **XML schemas** that may exist independently or that are generated from database data models.

Considerations

XML Schema Usage

- Search the **DoD Metadata Registry** for existing XML schemas suitable for reuse in system interfaces. Record the reuse of XML schemas in the DoD Metadata Registry and Clearinghouse.
- If an existing XML schema is close to but not exactly what was specified, review the system requirements with relevant **Communities of Interest** (**COIs**) to determine if the existing schema can be applied as-is or with minor modification.
- Review proposed XML definitions with the designated DoD XML Namespace Manager for relevant COIs.
- Define XML schemas only for that information for which the system is an authoritative source.
- Review XML definitions produced by government and industry consortia for possible reuse.
- Define XML interfaces in collaboration with known information exchange partners.

XML Schema Documentation

- Document the semantics of XML interfaces as annotations on the XML schema.
- Supply a text definition for every element, attribute, and enumeration value defined in the schema. Refer to the ***XML Schema*** specification [R1229] for more information on schema annotations.
- Describe the metadata for each **XML element** with information from related view, physical, logical, conceptual, and data element models.

Guidance

- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1390: Standardize on the terminology published by relevant **Communities of Interest** (**COIs**) listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**.

- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1724: Develop XML documents to be well formed.
- G1725: Develop XML documents to be **valid** XML.
- G1726: Define XML Schemas using **XML Schema Definition** (XSD).
- G1727: Provide names for XML type definitions.
- G1728: Define types for all **XML elements**.
- G1729: Annotate XML type definitions.
- G1737: Define a target namespace in schemas.
- G1738: Define a qualified namespace for the target namespace.
- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1759: Use a style guide when developing Web portlets.
- G1761: Provide units of measurements when displaying data.
- G1762: Indicate all simulated data as simulated.
- G1763: Indicate the security classification for all classified data.
- G1770: Explicitly define the **Data Distribution Service** (DDS) **Domains** for the system.
- G1796: Explicitly define all the **Data Distribution Service** (DDS) **Domain Topics**.
- G1798: Explicitly define all the **Data Distribution Service** (DDS) **Domain data types**.
- G1799: Explicitly associate data types to the **Data Distribution Service** (DDS) **Topics** within a DDS **Domain**
- G1800: Explicitly identify Keys within the **Data Distribution Service** (DDS) **data type** that uniquely identify an instance of a data object.
- G1810: Use **data models** to document the data contained within the **Data Distribution Service** (DDS) **Data-Centric Publish Subscribe** (DCPS).

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.

P1254: Design Tenet: Make Data Trustable

A key to supporting data trust relationships is to ensure that data is unchanged (or otherwise reconcilable) when the data is accessed from all points within the trust relationship. Formalize and enforce authoritative data sources and ensure that the data is current and distributed in a timely manner.

Considerations

- Use the Resource Descriptors and Security Descriptors specified by the **DoD Metadata Registry** to provide data validity and security information.
- Identify the authoritative source and purpose for each data element.
- Aggregated data can often exceed the security level of the individual data elements. Recognize and account for the possibility of an increased security level when aggregating data.

Guidance

- G1154: Use **stored procedures** for operations that are focused on the insertion and maintenance of data.
- G1155: Use **triggers** to enforce **referential** or **data integrity**, not to perform complex **business logic**.
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1762: Indicate all simulated data as simulated.
- G1763: Indicate the security classification for all classified data.

P1256: Design Tenet: Make Data Interoperable

To be interoperable, data must have known structural and discovery metadata as well as mechanisms to support its translation (e.g., to different units). Analyze and register metadata data assets such as names, data types, lengths, precision, scale, and restricted value domains. Identify the standards used to represent these items. Work with Communities of Interest to ensure the data represents appropriate semantics.

Considerations

XML Wrapped Data

- If XML wapped data are intended for exchange, configure them in terms of standard transactions with headers, trailers, and bodies.

XML Schema Validation

- Systems that produce XML documents shall guarantee that the XML documents are valid according to the XML schema they have published in the DoD Metadata Registry and Clearinghouse. Systems that receive XML documents should validate them against the schemas published by the Source system.

Guidance

- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1390: Standardize on the terminology published by relevant **Communities of Interest** (**COIs**) listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1724: Develop XML documents to be well formed.
- G1725: Develop XML documents to be **valid** XML.
- G1726: Define XML Schemas using **XML Schema Definition** (XSD).
- G1729: Annotate XML type definitions.
- G1737: Define a target namespace in schemas.
- G1738: Define a qualified namespace for the target namespace.
- G1746: Develop XSLT stylesheets that are XSLT version agnostic.

- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1754: Give each new XML schema version a unique URL.
- G1759: Use a style guide when developing Web portlets.
- G1761: Provide units of measurements when displaying data.
- G1763: Indicate the security classification for all classified data.
- G1770: Explicitly define the **Data Distribution Service** (DDS) **Domains** for the system.
- G1772: Assign a unique identifier for each **Data-Distribution Service** (DDS) **Domain** within the system.
- G1798: Explicitly define all the **Data Distribution Service** (DDS) **Domain data types**.
- G1799: Explicitly associate data types to the **Data Distribution Service** (DDS) **Topics** within a DDS **Domain**
- G1800: Explicitly identify Keys within the **Data Distribution Service** (DDS) **data type** that uniquely identify an instance of a data object.
- G1810: Use **data models** to document the data contained within the **Data Distribution Service** (DDS) **Data-Centric Publish Subscribe** (DCPS).
- G1796: Explicitly define all the **Data Distribution Service** (DDS) **Domain Topics**.
- G1001: Use formal standards to define public **interfaces**.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1865: Provide sufficient program, project, or initiative **metadata** descriptions and automated support to enable **mediation** and translation of the data between **interfaces**.
- BP1866: Coordinate with end users to develop interoperable materiel in support of high-value mission capability.

P1257: Design Tenet: Provide Data Management

Enhance the ability to support data management by providing a process to define, develop, and maintain an ontology (e.g., schemas, thesauruses, vocabularies, keyword lists, and taxonomies).

Considerations

- Obtain metrics to promote awareness of data management successes and areas requiring improvement.
- Provide a graphical representation, outline, or model representing the format, structure, and relationship of data.

Guidance

- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1390: Standardize on the terminology published by relevant **Communities of Interest** (**COIs**) listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1726: Define XML Schemas using **XML Schema Definition** (XSD).
- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1729: Annotate XML type definitions.
- G1647: Provide access to the **Federated Search** Services.
- G1125: Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1865: Provide sufficient program, project, or initiative **metadata** descriptions and automated support to enable **mediation** and translation of the data between **interfaces**.

Examples

- A database table and relationship structure
- A document type definition (DTD)

- A data structure used to pass information between systems
- An XML schema document (XSD) that represents a data structure and related information encoded as XML

P1258: Design Tenet: Be Responsive to User Needs

Include users in processes for creating discoverable, accessible, understandable, and trusted information and services. Understanding information interoperability creates an enviroment that can be responsive to users. User feedback mechanisms provide a means of capturing and reporting user satisfaction and give portfolio managers decision making information to steer investments, developments and improvements. Service and information providers in a mission area should work together to define the processes for using the user feedback for service and information improvements because these processes are specific to a portfolio of capabilities in the enterprise.

Considerations

- Provide a capability for capturing, tracking, and responding to user feedback.
- Collaborate with COIs in responding to user feedback.
- Ensure that user feedback is visible to the net-centric environment.
- Ensure that processes exist for consumers to do the following:
  - Request additional information from the information provider
  - Request changes in the format, i.e., syntax or semantics, of visible information
  - Report a problem with the information
- Establish metrics for determining responsiveness to user needs.

Guidance

- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1390: Standardize on the terminology published by relevant **Communities of Interest** (**COIs**) listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1391: Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.
- G1571: Maintain a comprehensive list of all the **Communities of Interest** (COIs) to which the **Components** of a Node belong.
- G1575: Designate Node representatives to relevant **Communities of Interest** (COIs) in which Components of the Node participate.
- G1760: Solicit feedback from users on user interface usability problems.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.
- BP1867: Use metrics to track responsiveness to user information sharing needs.

P1249: Services

A service is a contractually defined behavior a software component provides through a well-defined, published and shareable interface. The service concept is based on implementation characteristics like loose coupling, location independence, etc., that are inherently **net-centric**; this enables the rapid development and deployment of capabilities that, combined with other services, can provide a range of simple and complex functions that could be shared across diverse applications and management boundaries and woven into mission threads or business flows.

> ***Note:*** *For more information on service characteristics see Service-Oriented Architecture [P1304] in Part 1.*

Detailed Perspectives

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Scalability
Design Tenet: Availability
Design Tenet: Accommodate Heterogeneity
Design Tenet: Decentralized Operations and Management
Design Tenet: Enterprise Service Management

P1259: Design Tenet: Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is an architectural design style for building flexible, adaptable and distributed computing environments where functionality is exposed and shared across enterprise by the means of services.

> ***Note:*** *For more information on service-oriented architecture and service characteristics that enable the sharing of services across an enterprise see Service-Oriented Architecture [P1304] in Part 1.*

Considerations

Web Services

- Build Web services in accordance with the technical standards and conformance requirements prescribed by the current version of the WS-I Basic Profile. [R1237]
    - Use the WS-I Sample Application as a model for implementing and documenting Web services.
    - Use test tools authorized by WS-I that verify conformance with the current version of the WS-I Basic Profile.
    - Build and develop security extensions as prescribed in the current version of the WS-I Basic Security Profile.

Service Description

- Describe services using a standard Service Definition Framework (SDF). The SDF Perspective [P1296] provides a detailed specification for service definition and implementation. The SDF should address the following information for each service:
    - What the service does
    - How the service works (from a "black box" perspective)
    - Required security mechanisms or restrictions
    - Performance or quality of service (Q0S) information
    - Points of contact for the service
    - The specifics of how to bind to (access or use) the service

Service Access Point (SAP)

- Describe services provided by a system's SAPs. From a service provider perspective, SAPs can be abstracted away from the back-end or internal processing activities of the service. Looser coupling between SAP and service internals enables a service provider to change the internal workings of the back end, such as moving to a new version of a database, without changing the SAP.

Service Design

- Design services around operational requirements and service consumers' needs.
    - Base the service specifications on the needs of the initial users, since it is impossible to know all the possible service consumers.

- Provide an extensible interface so the service design can support future needs.

Service Design Characteristics

- Design services in accordance with best practices and patterns. For example, a service design should specify the information objects that are communicated across its interface in terms of enterprise metadata (e.g., time, location). These enable semantic agreement between the information objects.

- Design information objects to minimize the number of transactions across the service interface. An example of this is a request for an Authority to Operate (ATO), possibly constrained by a time and location attribute, followed by a reply containing the ATO that is applicable to a specific area of interest and time.

Service Implementation Characteristics

- Implementation information focuses on the technical implementation details that prospective service developers or providers need to design new services, or a service that uses another service. These attributes typically include items like the WSDL description of the service, details of a service's API interface point, and a description of service dependencies. Implement services using the following practices:

    - Document the open standards used.

    - Use vendor and platform independent messages.

    - Identify addresses using a Universal Resource Identifiers (URI).

    - Use defined and documented service interfaces.

    - Register XML interface descriptions using the DoD Metadata Registry and Clearinghouse.

    - Pass enterprise or COI objects, defined by their respective metadata, across its service interface.

    - Use extensible service interfaces with versioning, independent of the interface implementation version.

Service Level Agreement (SLA)

- Document a Service Level Agreement (SLA) to do the following:

    - Include quantitative measures for service usage, performance analysis, continuity of operations plan, and performance across the range of bandwidths provided by the node.

    - Have terms that the node's management services can monitor and manage.

    - Define responsibility for day-to-day service operations and procedures for reporting problems.

Service Interfaces

- Interface information should include descriptions of service features, service functionality, service provider identification, instructions on how to access and use the service through the SAP, and so on. The interface information should also discuss the different form factors that a service supports, such as a PDA.

- Express the Web service interfaces in **WSDL** in accordance with the current version of the WS-I Basic Profile.

- Register all XML schema files imported into WSDL under the appropriate namespace in the **DoD XML Registry**.

- At a minimum, store WSDL files in a file accessible via **URL** and **HTTP**.

Node Responsibilities for Services

- The node infrastructure should enable mission application software to be instantiated as services; this includes software libraries that support SOAP and WSDL processing. Node responsibilities include the following:
    - Using Web services standards (SOAP and WSDL) to interoperate applications across nodes.
    - Providing secure access to components in accordance with node and **GIG** IA/Security policies and services.
    - Designing services to be managed by the node in accordance with enterprise policy. Management services will typically be part of the node component framework environment (e.g., Java EE application server, .NET management environment) that is used in conjunction with NCES Enterprise Service Management.
    - Providing the capability to name and register components for local use within the node (e.g., **JNDI**). Component registration mechanisms shall interface or extend to service registration mechanisms, such as registration in the NCES Discovery service. If the component is only visible to the local node, it does not have to be registered in the NCES Discovery service.

Service Registration

- Systems register services using the standard service metadata in a directory available to the nodes in the enterprise. This directory may be based in the node, in an NCES Discovery Service, or both. At a minimum, identify a service by a Universal Resource Identifier (URI).
- Nodes register services as resources with the NCES Policy Management Service and control access to services using the NCES Policy Decision Services. The NCES Resource Attribute Services must provide access to service attributes.

Service Security

- Security information provides detailed information about the security specifications of the service, such as restrictions on who can use or access the service, for example indicating that the user must present a valid DoD PKI certificate to access the service.
- A security framework is required at the node level to authenticate principals, ensure confidentiality and integrity of messages and authorize access.
- Use security mechanisms provided by the node. These must include mutal authentication over an encrypted channel such as SSL, authorization, confidentiality, integrity and non-repudiation.
- Services must support **role-based access control** (**RBAC**) mechanisms.
- Nodes should provide interfaces to NCES security services.
- Nodes should establish trust relationships with other nodes in the enterprise using the NCES Domain Federation Service.

Support for Service Orchestration

- Provide the capability to compose mission capabilities from one or more services using a service orchestration or workflow mechanism based on industry standards such as BPEL.

Guidance

- G1001: Use formal standards to define public **interfaces**.
- G1002: Separate public **interfaces** from implementation.

- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.
- G1004: Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.
- G1005: Separate infrastructure capabilities from **mission** functions.
- G1007: Ensure that applications use open, standardized, **vendor**-neutral **API**(s).
- G1008: Isolate platform-specific **interfaces** and **vendor** dependencies.
- G1010: Use **open-standard** logging frameworks.
- G1011: Make components independently deployable.
- G1012: Use a set of services to expose **Component** functionality.
- G1014: Access databases through **open standard** interfaces.
- G1018: Assign version identifiers to all public interfaces.
- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1021: Create fully insulated classes.
- G1022: Insulate public **interfaces** from compile-time dependencies.
- G1027: Internally document all source code developed with DoD funding.
- G1030: Use a standard GUI **component** library.
- G1032: Validate all input fields.
- G1035: Follow W3C standards for code which will generate a Web page display.
- G1043: Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.
- G1044: Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.
- G1045: Define **XML** format information separately in **XSL**.
- G1050: In **ASP**, isolate the presentation tier from the middle tier using **COM** objects.
- G1052: Use the code-behind feature in ASP.NET to separate presentation code from the business logic.
- G1053: Do not embed HTML code in any code-behind code used by aspx pages.
- G1056: Specify a versioning policy for **.NET** assemblies.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1060: Encapsulate Java code that is used in **JSP**(s) in tag libraries.
- G1071: Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).
- G1073: Isolate vendor extensions to enterprise-services standard interfaces.
- G1078: Document the use of non-**Java EE**-defined **deployment descriptors**.

- G1079: Isolate tailorable data values into the **deployment descriptors** for **Java EE** applications.
- G1080: Adhere to the **Web Services Interoperability Organization** (**WS-I**) Basic Profile specification for **Web service** environments.
- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1083: Do not pass **Web Services-Interoperability Organization** (**WS-I**) **Document Object Model** (**DOM**) documents as strings.
- G1084: Validate documents transferred using **SOAP** against the **W3C XML** Standard by an **XML Schema Definition** (**XSD**) defined by the **Community of Interest** (**COI**).
- G1085: Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.
- G1087: Validate all **Web Services Definition Language** (**WSDL**) files that describe **Web services**.
- G1088: Use isolation design patterns to define system functionality that manipulates **Web services**.
- G1090: Do not **hard-code** a **Web service's endpoint**.
- G1093: Implement exception handlers for **SOAP**-based **Web services**.
- G1095: Use **W3C** fault codes for all **SOAP** faults.
- G1101: Use **Web services** to bridge **Java EE** and **.NET**.
- G1118: Localize **CORBA** vendor-specific source code into separate **modules**.
- G1119: Isolate user-modifiable configuration parameters from the **CORBA** application source code.
- G1121: Do not modify **CORBA** Interface Definition Language (**IDL**) compiler auto-generated stubs and skeletons.
- G1123: Use the Fat Operation Technique in **IDL** operator invocation.
- G1125: Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.
- G1127: Use a **UDDI** specification that supports publishing discovery services.
- G1131: Use industry standard Universal Description, Discovery, and Integration (**UDDI**) **APIs** for all UDDI inquiries.
- G1132: Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement the **SQL** standard.
- G1141: Use standard **data models** developed by **Communities of Interest** (**COI**) as the basis of program or project data models.
- G1144: Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.
- G1146: Include information in the **data model** necessary to generate a **data dictionary**.
- G1147: Use **domain analysis** to define the constraints on input data validation.
- G1148: **Normalize** data models.
- G1151: Define declarative **foreign keys** for all relationships between tables to enforce **referential integrity**.

- G1153: Separate application, presentation, and data tiers.
- G1154: Use **stored procedures** for operations that are focused on the insertion and maintenance of data.
- G1155: Use **triggers** to enforce **referential** or **data integrity**, not to perform complex **business logic**.
- G1190: Use a build tool.
- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1204: Create configuration services to provide distributed user control of the appropriate configuration parameters.
- G1205: Use non-source code persistence to store all user-modifiable **CORBA** service configuration parameters.
- G1208: Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.
- G1209: For Java, use **JDK** logging facilities.
- G1210: For **.NET**, use Debug and Trace from the **`System.Diagnostics`** **namespace**.
- G1217: Develop and use externally configurable components.
- G1218: Use a build tool that supports operation in an automated mode.
- G1219: Use a build tool that checks out files from configuration control.
- G1220: Use a build tool that **compiles** source code and dependencies that have been modified.
- G1221: Use a build tool that creates libraries or archives after all required compilations are completed.
- G1222: Use a build tool that creates executables.
- G1223: Use a build tool that is capable of running unit tests.
- G1224: Use a build tool that cleans out intermediate files that can be regenerated.
- G1225: Use a build tool that is independent of the **Integrated Development Environment**.
- G1236: Do not **hard-code** the **endpoint** of a **Web service** vendor.
- G1237: Do not **hard-code** the configuration data of a **Web service** vendor.
- G1239: Use design patterns (e.g., **facade**, **proxy**, or **adapter**) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.
- G1245: Isolate the **Web service** **portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.
- G1267: Use industry standard HTML data entry fields on Web pages.
- G1268: Label all data entry fields.
- G1270: Include scroll bars for text entry areas if the data buffer is greater than the viewable area.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1276: Do not modify the contents of the Web browser's status bar.

- G1277: Do not use tickers on a Web site.
- G1278: Use the browser default setting for links.
- G1283: Use **linked style sheets** rather than embedded styles.
- G1284: Use only one font for **HTML** body text.
- G1285: Use **relative font sizes**.
- G1286: Provide text labels for all buttons.
- G1287: Provide feedback when a transaction will require the user to wait.
- G1292: Use text-based Web site navigation.
- G1293: Use descriptive labels for all clickable graphics.
- G1294: Provide a site map on all Web sites.
- G1295: Provide redundant text links for images within an **HTML** page.
- G1356: Use the **SOAP** standard for all **Web services**.
- G1566: Use `alt` attributes to provide alternate text for non-text items such as images.
- G1569: Maintain a comprehensive list of all of the **Components** that are part of the Node.
- G1573: Define the enterprise design patterns that a Node supports.
- G1574: Define which enterprise design patterns a **Component** requires.
- G1579: Define which **Enterprise Services** the Node will host locally when the Node becomes operational.
- G1580: Define which **Enterprise Services** will be hosted over the **Global Information Grid** (GIG) when the Node becomes operational.
- G1581: Expose **legacy system** or **application** functionality through the use of a service that uses a **facade design pattern**.
- G1635: Make Nodes that will be part of the **Global Information Grid** (GIG) consistent with the *GIG Integrated Architecture*.
- G1636: Comply with the **Net-Centric Operations and Warfare Reference Model** (NCOW RM).
- G1637: Make Node-implemented **directory services** comply with the directory services **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs).
- G1638: Comply with the directory services **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node directory services **proxies**.
- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.
- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.
- G1641: Comply with the Service Discovery **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node-implemented **Service Discovery** (SD).

- G1642: Comply with the **Service Discovery Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node Service Discovery (SD) **proxies**.

Best Practices

- BP1863: Make shareable data assets visible, even if they are not accessible.
- BP1689: Use the **Service Discovery** (SD) pilot program to practice and exercise the mechanics of service discovery and late binding.

P1268: Design Tenet: Open Architecture

Design mission application software to be separable from the supporting **node** and to access the node through public interfaces based on standards governed by a recognized standards organization (e.g., IEEE, W3C, OASIS).

Considerations

Component Based

- Architect mission application software in the node as components integrated within a node. Provide run-time and resource management services (e.g., component management, security, virtual machines, memory management, object management, resource pooling).
- Include component frameworks in the node based on commercially available solutions without proprietary extensions. Wrap any extensions, if used, via the appropriate design pattern.
- Architect and manage mission application software that spans multiple nodes in a manner that aligns with all of the supporting nodes.

Note: *Examples include **Java Platform, Enterprise Edition** (**Java EE**), **Common Object Request Broker Architecture** (**CORBA**), **.NET Framework**, and **Data Distribution System** (**DDS**).*

Public Interfaces

- Provide the mechanism on the node for components to expose public interfaces. The interface must be separate from the implementation. Base the public interface mechanism on the node component framework. These public interfaces must be visible to other components in the node.

Layered Software Architecture

- Layer application software using an N-tier architecture. At a minimum, use discrete ***client***, ***presentation***, ***middle***, and ***data*** tiers.
  - ***Client Tier*** -The client tier supports a wide range of device types such as desktop computers, laptops, mobile, wireless, and personal digital assistant (PDA). It supports direct interaction with the user.
  - ***Presentation Tier -*** The presentation tier provides content to a range of client device types supported by the node (e.g., Hypertext, eXtensible or Wireless Markup Language [**HTML**, **XML**, **WML**]). Implement presentation components with the mechanisms in the node's component framework.
  - ***Middle Tier -*** The middle tier supports the construction of componentized business logic and public interfaces (e.g., interface classes). Base business components on programming mechanisms provided by the component framework chosen by the node (e.g. **Enterprise Java Beans**, CORBA services, **COM** components). Specific business logic elements, such as data validation, may reside in other tiers.
  - ***Data Tier -*** Base access to the data tier within nodes on industry open-standard mechanisms such a **SQL** or **JDBC**/**ODBC**. Use services to access data across nodes.

Wrapping Legacy Systems

- Wrap legacy application software with an interface that is accessible from the node; for example, use Java Connector Architecture on a Java EE platform. See ***Part 3: Migration Guidance*** (e.g., Pattern: Wrapping Legacy Code into a Service) for additional information on wrapping legacy systems.

Guidance

- G1001: Use formal standards to define public **interfaces**.
- G1002: Separate public **interfaces** from implementation.
- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.
- G1004: Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.
- G1005: Separate infrastructure capabilities from **mission** functions.
- G1007: Ensure that applications use open, standardized, **vendor**-neutral **API**(s).
- G1008: Isolate platform-specific **interfaces** and **vendor** dependencies.
- G1010: Use **open-standard** logging frameworks.
- G1011: Make components independently deployable.
- G1012: Use a set of services to expose **Component** functionality.
- G1014: Access databases through **open standard** interfaces.
- G1018: Assign version identifiers to all public interfaces.
- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1021: Create fully insulated classes.
- G1022: Insulate public **interfaces** from compile-time dependencies.
- G1027: Internally document all source code developed with DoD funding.
- G1030: Use a standard GUI **component** library.
- G1032: Validate all input fields.
- G1035: Follow W3C standards for code which will generate a Web page display.
- G1043: Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.
- G1044: Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.
- G1045: Define **XML** format information separately in **XSL**.
- G1050: In **ASP**, isolate the presentation tier from the middle tier using **COM** objects.
- G1052: Use the code-behind feature in ASP.NET to separate presentation code from the business logic.
- G1053: Do not embed HTML code in any code-behind code used by aspx pages.
- G1056: Specify a versioning policy for **.NET** assemblies.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1060: Encapsulate Java code that is used in **JSP**(s) in tag libraries.
- G1071: Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).

- G1073: Isolate vendor extensions to enterprise-services standard interfaces.
- G1078: Document the use of non-**Java EE**-defined **deployment descriptors**.
- G1079: Isolate tailorable data values into the **deployment descriptors** for **Java EE** applications.
- G1080: Adhere to the **Web Services Interoperability Organization** (**WS-I**) Basic Profile specification for **Web service** environments.
- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1083: Do not pass **Web Services-Interoperability Organization** (**WS-I**) **Document Object Model** (**DOM**) documents as strings.
- G1084: Validate documents transferred using **SOAP** against the **W3C XML** Standard by an **XML Schema Definition** (**XSD**) defined by the **Community of Interest** (**COI**).
- G1085: Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.
- G1087: Validate all **Web Services Definition Language** (**WSDL**) files that describe **Web services**.
- G1088: Use isolation design patterns to define system functionality that manipulates **Web services**.
- G1090: Do not **hard-code** a **Web service's endpoint**.
- G1093: Implement exception handlers for **SOAP**-based **Web services**.
- G1095: Use **W3C** fault codes for all **SOAP** faults.
- G1101: Use **Web services** to bridge **Java EE** and **.NET**.
- G1118: Localize **CORBA** vendor-specific source code into separate **modules**.
- G1119: Isolate user-modifiable configuration parameters from the **CORBA** application source code.
- G1121: Do not modify **CORBA** Interface Definition Language (**IDL**) compiler auto-generated stubs and skeletons.
- G1123: Use the Fat Operation Technique in **IDL** operator invocation.
- G1125: Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.
- G1127: Use a **UDDI** specification that supports publishing discovery services.
- G1131: Use industry standard Universal Description, Discovery, and Integration (**UDDI**) **APIs** for all UDDI inquiries.
- G1132: Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement the **SQL** standard.
- G1141: Use standard **data models** developed by **Communities of Interest** (**COI**) as the basis of program or project data models.
- G1144: Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.
- G1153: Separate application, presentation, and data tiers.
- G1190: Use a build tool.

- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1204: Create configuration services to provide distributed user control of the appropriate configuration parameters.
- G1205: Use non-source code persistence to store all user-modifiable **CORBA** service configuration parameters.
- G1208: Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.
- G1209: For Java, use **JDK** logging facilities.
- G1210: For **.NET**, use Debug and Trace from the **`System.Diagnostics`** **namespace**.
- G1213: Provide an architecture design document.
- G1214: Provide a document with a plan for **deprecating** obsolete **interfaces**.
- G1215: Provide a coding standards document.
- G1216: Provide a software release plan document.
- G1217: Develop and use externally configurable components.
- G1218: Use a build tool that supports operation in an automated mode.
- G1219: Use a build tool that checks out files from configuration control.
- G1220: Use a build tool that **compiles** source code and dependencies that have been modified.
- G1221: Use a build tool that creates libraries or archives after all required compilations are completed.
- G1222: Use a build tool that creates executables.
- G1223: Use a build tool that is capable of running unit tests.
- G1224: Use a build tool that cleans out intermediate files that can be regenerated.
- G1225: Use a build tool that is independent of the **Integrated Development Environment**.
- G1236: Do not **hard-code** the **endpoint** of a **Web service** vendor.
- G1237: Do not **hard-code** the configuration data of a **Web service** vendor.
- G1239: Use design patterns (e.g., **facade**, **proxy**, or **adapter**) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.
- G1245: Isolate the **Web service** **portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.
- G1267: Use industry standard HTML data entry fields on Web pages.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1276: Do not modify the contents of the Web browser's status bar.
- G1278: Use the browser default setting for links.
- G1284: Use only one font for **HTML** body text.

- G1285: Use **relative font sizes**.
- G1356: Use the **SOAP** standard for all **Web services**.
- G1573: Define the enterprise design patterns that a Node supports.
- G1574: Define which enterprise design patterns a **Component** requires.
- G1581: Expose **legacy system** or **application** functionality through the use of a service that uses a **facade design pattern**.
- G1626: Identify which **Core Enterprise Services** (CES) capabilities the Node **Components** require.
- G1627: Identify the priority of each **Core Enterprise Services** (CES) capability the Node **Components** require.
- G1629: Identify which **Net-Centric Enterprise Services** (NCES) capabilities the Node requires during deployment.
- G1630: Comply with the applicable **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) for implemented **Core Enterprise Services** (CES) in the Node.
- G1631: Expose **Core Enterprise Services** (CES) that comply with the applicable **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in all Node services **proxies**.
- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.
- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.
- G1724: Develop XML documents to be well formed.
- G1725: Develop XML documents to be **valid** XML.
- G1726: Define XML Schemas using **XML Schema Definition** (XSD).
- G1727: Provide names for XML type definitions.
- G1728: Define types for all **XML elements**.
- G1729: Annotate XML type definitions.
- G1737: Define a target namespace in schemas.
- G1738: Define a qualified namespace for the target namespace.
- G1746: Develop XSLT stylesheets that are XSLT version agnostic.
- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1754: Give each new XML schema version a unique URL.
- G1770: Explicitly define the **Data Distribution Service** (DDS) **Domains** for the system.

Best Practices

- BP1863: Make shareable data assets visible, even if they are not accessible.

- BP1864: Layer architectures to support clear boundaries between data management, presentation, and business logic functionality.

P1270: Design Tenet: Scalability

Design services and components to use resource management mechanisms that the hosting Node provides to enable scalability under load. For example, use buffer and connection pools, tuned to the expected user load, to enable concurrent user sessions with acceptable performance.

- Scalability is the extent to which the organization, program, project, or initiative can grow to accommodate additional users. Scalable components are either co-located or globally distributed. Scalability of computing infrastructure (CI) components and CI-related doctrine, organization, training, materiel, leadership and education, personnel, and facilities (DOTMLPF) allows for rapidly implemented increases in capacity and capability to support program, project, and initiative growth or dynamically changing requirements.

To the greatest extent possible given bandwidth and technical environment considerations, make services accessible in an open-systems, interface-driven, distributed computing environment with reusable components available to the enterprise. Acceptable Web-based methods are represented by Internet standards and protocols registered in the **Defense IT Standards Registry** (**DISR**) and managed by the DoD IT Standards Committee (ITSC). To the greatest extent possible, the service design should include considerations for potential edge users with limited bandwidth access and limited display or storage capacity. As enterprise services emerge, the infrastructure should establish new parameters related to maintainability, scalability, performance, orchestration, accreditation, and availability.

Considerations

Design Factors

- System architects, program managers, and designers for a program, project or initiative should consider a vision that includes growth projections for the program's foreseeable future.

Assessing Scalability Requirements

- Assess and evaluate requirements and capabilities of services to understand scalability hot spots better
- Properly estimate usage patterns
- Manage user authentication/authorization
- Manage session state where applicable
- Scale user or internal facing Web sites
- Scale data resources
- Scale CPU load

Stateless Service

- Each message that a consumer sends to a provider must contain all necessary information for the provider to process it. This constraint makes a service provider more scalable because the provider does not have to store state information between requests.

Stateful Service

- Stateful service is difficult to avoid in a number of situations. For example, establishing a session between a consumer and a provider for efficiency reasons such as sending a security certificate with each request. The process creates a load for both consumer and provider. It is much quicker to replace the certificate with a token shared just between the consumer and provider. Stateful services require both the consumer and the provider to share the same consumer-specific context, which is either included in or referenced by messages exchanged between the provider and the consumer. The problem with this constraint is that it potentially reduces the overall scalability of the service. The service provide must remember context for each consumer. Coupling between a service provider and a consumer is increased. Switching service providers is more difficult.

Guidance

- G1012: Use a set of services to expose **Component** functionality.
- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1088: Use isolation design patterns to define system functionality that manipulates **Web services**.
- G1123: Use the Fat Operation Technique in **IDL** operator invocation.
- G1153: Separate application, presentation, and data tiers.
- G1283: Use **linked style sheets** rather than embedded styles.
- G1352: Use database clustering and redundant array of independent disks (RAID) for high availability of data.
- G1572: Include the Node as a party to any **Service Level Agreements** (SLAs) signed by any of the **Components** of the Node.

Best Practices

- BP1864: Layer architectures to support clear boundaries between data management, presentation, and business logic functionality.

P1271: Design Tenet: Availability

As the net-centric environment evolves, an ever increasing number of information services will become available to DoD users. At the same time, infrastructure support for these services will also transform to net-centric standards # leveraging shared processing and storage on the GIG and dynamic allocation. It will be critical in this environment to maintain acceptable and measurable levels of support for all **enterprise** capabilities. When users seek, find and use an **Enterprise Service**, they will have certain expectations regarding its pedigree, reliability and availability. These attributes should be consistent across all Enterprise Services.

Design services and components to meet the availability requirements of the node. The implementation should use the maintenance strategies and management mechanisms provided by the Node's infrastructure.

Considerations

- While an Enterprise Service may be provided from anywhere in the **Global Information Grid** (**GIG**), user expectations demand that they be hosted in environments that meet minimum GIG computing node standards in terms of availability, support and backup.

Guidance

- G1352: Use database clustering and redundant array of independent disks (RAID) for high availability of data.

- G1572: Include the Node as a party to any **Service Level Agreements** (SLAs) signed by any of the **Components** of the Node.

Best Practices

- BP1868: Incorporate mechanisms to enhance the survivability, resiliency, redundancy, and reliability of Computing Infrastructure (CI).

P1275: Design Tenet: Accommodate Heterogeneity

Th **Global Information Grid** (**GIG**) is a heterogeneous environment. No one product will meet the needs of potentially vastly different operational environments. Services and Service-Oriented Architecture (SOA) related infrastructure will need to interoperate across these diverse environments.

Considerations

Service Structure

- Design systems to be able to deploy services separately from the supporting **node**. The services should access the node through public interfaces.

Service Configuration

- Design systems to be able to configure services on each node on which they are deployed. Use external configuration file mechanisms (e.g., deployment descriptors for **Java EE** applications) to specify the configuration. Do not use hard-coded configuration parameters that require a binary tool to update or that require a recompile and relink.

Node Structure

- Nodes provide the infrastructure and rules for assembling, configuring, deploying, securing, operating, and managing mission applications and services. For more information, see NESI Part 4: Node Guidance [P1130].
- Nodes are responsible for provisioning their diverse mission application and services. They must configure and operate them in accordance with enterprise management policy.

Guidance

- G1001: Use formal standards to define public **interfaces**.
- G1002: Separate public **interfaces** from implementation.
- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.
- G1004: Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.
- G1005: Separate infrastructure capabilities from **mission** functions.
- G1007: Ensure that applications use open, standardized, **vendor**-neutral **API**(s).
- G1008: Isolate platform-specific **interfaces** and **vendor** dependencies.
- G1010: Use **open-standard** logging frameworks.
- G1011: Make components independently deployable.
- G1012: Use a set of services to expose **Component** functionality.
- G1014: Access databases through **open standard** interfaces.
- G1018: Assign version identifiers to all public interfaces.

- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1021: Create fully insulated classes.
- G1022: Insulate public **interfaces** from compile-time dependencies.
- G1030: Use a standard GUI **component** library.
- G1032: Validate all input fields.
- G1035: Follow W3C standards for code which will generate a Web page display.
- G1043: Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.
- G1044: Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.
- G1045: Define **XML** format information separately in **XSL**.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1071: Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).
- G1073: Isolate vendor extensions to enterprise-services standard interfaces.
- G1080: Adhere to the **Web Services Interoperability Organization** (**WS-I**) Basic Profile specification for **Web service** environments.
- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1083: Do not pass **Web Services-Interoperability Organization** (**WS-I**) **Document Object Model** (**DOM**) documents as strings.
- G1084: Validate documents transferred using **SOAP** against the **W3C XML** Standard by an **XML Schema Definition** (**XSD**) defined by the **Community of Interest** (**COI**).
- G1087: Validate all **Web Services 4993 Tm [L3] Tsu( ) (rest)] TJ /F1 10 Tf 0 g 1 0 0 -10436.969997 452.554993 Tm [( )**

- G1141: Use standard **data models** developed by **Communities of Interest** (**COI**) as the basis of program or project data models.
- G1153: Separate application, presentation, and data tiers.
- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1204: Create configuration services to provide distributed user control of the appropriate configuration parameters.
- G1208: Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.
- G1209: For Java, use **JDK** logging facilities.
- G1210: For **.NET**, use Debug and Trace from the `System.Diagnostics` **namespace**.
- G1217: Develop and use externally configurable components.
- G1236: Do not **hard-code** the **endpoint** of a **Web service** vendor.
- G1237: Do not **hard-code** the configuration data of a **Web service** vendor.
- G1239: Use design patterns (e.g., **facade**, **proxy**, or **adapter**) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.
- G1245: Isolate the **Web service portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.
- G1267: Use industry standard HTML data entry fields on Web pages.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1276: Do not modify the contents of the Web browser's status bar.
- G1278: Use the browser default setting for links.
- G1284: Use only one font for **HTML** body text.
- G1285: Use **relative font sizes**.
- G1292: Use text-based Web site navigation.
- G1293: Use descriptive labels for all clickable graphics.
- G1295: Provide redundant text links for images within an **HTML** page.
- G1566: Use `alt` attributes to provide alternate text for non-text items such as images.
- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.
- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.

Best Practices

- BP1864: Layer architectures to support clear boundaries between data management, presentation, and business logic functionality.

- BP1870: Conform to DoD-specified data publication methods that are consistent with **Global Information Grid** (**GIG**) enterprise and user technologies per DoD Directive 8101.1. [R1166]

P1276: Design Tenet: Decentralized Operations and Management

Design services to provide a management interface that either the **node's** management services or the **Net-Centric Enterprise Services** (**NCES**) Enterprise Service Management services can access. Intuitive management interfaces provide operators with the toolset to be responsive to system operations, system changes, and maintenance needs. Design management interfaces that new personnel can easily learn with minimum training to mitigate loss of knowledge and skill sets caused by troop rotation or personnel turnover. Use **COTS** products with Web-based **GUIs** that enable operators or administrators to make configuration changes easily, execute maintenance utilities (e.g., log capture, backups), check operational performance/status, and facilitate user administration.

Considerations

- Support a decentralized operational concept where other systems, services, or capabilities are providing key elements of the end-to-end **net-centric** solution.
- Provide an integrated digital environment to enhance communications and productivity for management and operations of programs, projects or initiatives.
- Provide remote management capabilities that are employed to manage the distributed computing infrastructure such as Telnet, Secure Shell, Web-based proprietary, Web-based COTS or customized COTS, or other technologies.
- Provide security and access control mechanisms to facilitate management across differing security domains in the DoD, Intelligence Community, other government agencies, and coalition partners.

Guidance

- G1204: Create configuration services to provide distributed user control of the appropriate configuration parameters.
- G1606: Manage **routers** remotely from within the **Node**.
- G1347: Secure remote connections to a database.
- G1623: Implement personal **firewall** software on **client** or **server** hardware used for remote connectivity in accordance with the Desktop Applications, Network and Enclave **Security Technical Implementation Guides** (**STIGs**).
- G1245: Isolate the **Web service** **portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.

P1278: Design Tenet: Enterprise Service Management

Considerations

Service Management

- Service management includes tracking the development, deployment, and operation of services. Manage services according to **Node** affiliation using available management services, either **NCES** Enterprise Service Management or local services.
- Expose a service management interface that the node management services can access.

Provisioning of Enterprise Services

- Design the Node's applications and components to enable access to enterprise services as they become available from DoD/DISA.
- When required, implement enterprise services locally at the Node based on technical standards provided by DoD/DISA. When such standards are not specified, choose standards based on best commercial practice.
- Maintain a separable service implementation to enable the replacement of local Node implementations with NCES services as they become available.

Guidance

- G1010: Use **open-standard** logging frameworks.
- G1032: Validate all input fields.
- G1093: Implement exception handlers for **SOAP**-based **Web services**.
- G1094: Catch all exceptions for application code exposed as a **Web service**.
- G1095: Use **W3C** fault codes for all **SOAP** faults.
- G1132: Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement the **SQL** standard.
- G1155: Use **triggers** to enforce **referential** or **data integrity**, not to perform complex **business logic**.
- G1209: For Java, use **JDK** logging facilities.
- G1210: For **.NET**, use Debug and Trace from the `System.Diagnostics` **namespace**.
- G1276: Do not modify the contents of the Web browser's status bar.
- G1287: Provide feedback when a transaction will require the user to wait.
- G1639: Describe **Components** exposed by the Node as specified by the **Service Definition Framework**
- G1569: Maintain a comprehensive list of all of the **Components** that are part of the Node.

Best Practices

- BP1868: Incorporate mechanisms to enhance the survivability, resiliency, redundancy, and reliability of Computing Infrastructure (CI).

P1240: Information Assurance/Security

Information assurance (IA) refers to measures that protect and defend information and information systems. The goal of IA is to ensure confidentiality, integrity, availability, and accountability by providing capabilities to detect, monitor, react to, and protect against attacks.

Many of the existing solutions to IA problems (and many of the requirements in existing IA regulations) assume that both clients and servers are located on the same physical or logical network. They rely heavily on perimeter or boundary protection. Service-oriented architecture (SOA) interoperability and loose coupling requirements make those security models inadequate.

In SOA, the boundaries are not clearly defined. Services may be exposed to external clients and not bound to a physical location. The client and service providers may be governed by different security policies.

Base a net-centric IA strategy on a service-level view of security rather than on perimeter security. Developing new security models is necessary to determine how to establish the necessary trust relationships between service requestors and service providers and to select the most adequate and appropriate authentication and authorization mechanisms. To implement a net-centric IA strategy, programs should provide the following:

- Integrated identity management, permissions management, and digital rights management
- Adequate confidentiality, availability, and integrity

Detailed Perspectives

P1242: Design Tenet: Net-Centric IA Posture and Continuity of Operations

This tenet refers to the assignment of Mission Assurance Category (MAC) and Confidentiality Level to a given application, node, or system. The MAC reflects the importance of information relative to the achievement of DoD goals and objectives, particularly the warfighter's combat mission. Mission Assurance Categories primarily determine the requirements for availability and integrity.

There are three defined mission assurance categories:

- MAC I for systems with vital operational needs
- MAC II for systems that are important to deployed or contingency forces
- MAC III for systems supporting day-to-day businesses that do not materially affect support to deployed forces

The complete definitions for those categories are included in DoD Directive 8500.1. [R1197] The security requirement for each combination of mission assurance category and its confidentiality level are in DoD Instruction 8500.2. [R1198]

Considerations

- When assigning a MAC in a net-centric environment, consider not just the intrinsic properties of the node or service, but also its impact on other Information Operations that may call upon it.
- When developing a node or service, account for its potential use by other missions and adjust the MAC appropriately. Incorporate adequate protection and integrity requirements into the design that are commensurate with those potential uses.
- Typically, not all of the potential uses of a node or service are known up front. Therefore, developers must make assumptions about how critical missions may use the node or service when they determine requirements. It may be necessary to modify the MAC to accommodate future, critical missions.

Guidance

- G1634: Certify and accredit **Components** with all applicable DoD **Information Assurance** (IA) processes.
- G1632: Certify and accredit Nodes with all applicable DoD **Information Assurance** (IA) processes.
- G1633: Host only DoD **Information Assurance** (IA) certified and accredited **Components**.
- G1585: Provide a transport infrastructure for the Node that implements **Global Information Grid** (GIG) **Information Assurance** (IA) boundary protections.

Best Practices

- BP1701: Configure **Components** for **Information Assurance** (IA) in accordance with the Network **Security Technical Implementation Guide** (STIG).
- BP1672: Be prepared to integrate fully with the **Information Assurance** (IA) infrastructure.

P1243: Design Tenet: Identity Management, Authentication, and Privileges

Authentication mechanisms are based on **credentials** presented by the requestor. Those credentials may be something the user knows (e.g., passwords), something the user is (e.g., biometrics), something the user has (e.g., smart card), or any combination of these factors.

Each approach is associated with the strength of an authentication. The weakest methods are password-based and the strongest are combinations of biometrics and smart cards.

There are also differing strengths within each method. For instance, systems that require complex passwords are stronger than those that accept simple ones and systems using retina or fingerprint readers are stronger than those that use finger length.

Components that are separate from the implementation of mission- or business-specific functionality often provide identity authentication management and authorization.

Considerations

User Authentication

- Authentication normally occurs at the "edge" of an application or node, or at the very first network access. Systems should strive to accept strong authentication methods as early as possible. If possible, migrate authentication tasks to an authentication server and make systems rely on tokens or assertions from the server for authentication. For closed community configurations, these schemes may involve the use of a Kerberos-type single sign-on device.

Identity Management

- Use authentication assertions to propagate identities in a secure and trusted way throughout the enterprise. Those assertions should indicate not only the identity and attributes of the requestor, but the strength of the mechanism used to ascertain its identity.

- Generate a Trust Model to specify the proper trust relationships and the path for authentication assertions.

Multi-Tier Authentication

- While considering the specific method used and its relative strength, remember that in a **Service-Oriented Architecture** (**SOA**) service providers may require stronger authentication than that invoked by the service requestor. These cases may require a multi-tier authentication; i.e., re-authenticating the original requester with the provider by transferring appropriate credentials.

- To avoid future multi-tier authentication problems, use strong authentication methods such as PKI certificates whenever possible.

Authorization Processes

- Access authorizations are determined by the requester's attributes and by the nature and contents of the request. Make the authorization decision at the access boundary, isolating the application from changes in policy and authorization technology.

- Use node-managed security (sometimes referred to as declarative security, programmatic security, or container-managed security), unless application requirements require programmatic authorizations, where individual actions within the service are authorized based on the nature or parameters of the request.

Role-Based Authorizations

- Roles are one way to establish authorized access control. In the **Role-Based Access Control** (**RBAC**) environment, role privileges are the basis for access decisions. In RBAC, a trusted entity administers users and their roles in association with the user identity. Users should never supply a mapping of users to roles directly, but users may select one of multiple roles assigned to them when seeking access to system functionality. Use the eXtensible Access Control Markup Language (XACML) to retrieve access control information. XACML supports the exchange of access control information using XML. This allows adherence to the principle of least privilege (see the following perspective for additional information on this principle: Apply Principle of Least Privilege [P1317]).

Attribute-Based Authorizations

- Attribute-Based Access Control (ABAC) isolates the service provider from changes in the user population. In the ABAC environment, a set of user attributes is the basis for access decisions. These attributes could include, for example, level of clearance, level of training, and specific assignment location.

- When an application retrieves access control information from an external policy decision point or retrieves policies for its own resources, it should do so with XACML which supports exchange of access control information using XML. In general, authorization policies should be distinct from application functionality but co-located and co-managed with those applications.

Validation of Authentication Information

- A service provider may receive requests that include the original authentication information from the requestor. DoD uses Public Key Infrastructure (PKI) certificates for authentication information. A very effective way for the provider to ascertain the validity of the authentication information is to confirm it through a PKI mechanism.

- A service provider, when receiving requestor identification information through a security assertion, must authenticate that an entity that the provider trusts has validated the assertion. PKI signatures provide a means to accomplished this. The signatures must encompass and link both the assertion and the actual request. The service provider must determine, if using PKI, the complete scheme of how to verify the certificates, the timeliness of the requests, and the current validity of the credential (i.e., verification that the certificates are revoked).

- Systems should migrate to PKI authentication as it become available, and start using it as a baseline to provide enterprise authentication services.

Guidance

- G1300: Secure all **endpoints**.

- G1302: Validate all inputs.

- G1306: **Identify** and **authenticate** users of the application.

- G1307: Provide a security policy file.

- G1308: Configure **Public Key Enabled** applications to use a **Federal Information Processing Standard** (**FIPS**) 140-2 certified cryptographic module.

- G1309: Make applications handling high value unclassified information in Minimally Protected environments **Public Key Enabled** to interoperate with **DoD High Assurance** .

- G1310: Protect application cryptographic objects and functions from tampering.

- G1311: Use **Hypertext Transfer Protocol over Secure Socket Layer** (**HTTPS**) when applications communicate with DoD **Public Key Infrastructure** (**PKI**) components.

- G1312: Make applications capable of being configured for use with DoD **PKI**.
- G1313: Provide documentation for application configuration and setup for use with DoD **PKI**.
- G1314: Provide applications the ability to import and export keys (software certificates only).
- G1315: For applications, use key pairs and **Certificates** created for individuals using DoD **PKI** methods and procedures defined by the DoD Class 3 Public Key Infrastructure Interface Specification and the Personal Information Exchange Syntax Standard.
- G1316: Ensure that applications protect **private keys**.
- G1317: Ensure applications store **Certificates** for subscribers (the owner of the **Public Key** contained in the Certificate) when used in the context of signed and/or encrypted email.
- G1318: Develop applications such that they provide the capability to manage and store **trust points** (**Certificate Authority** Public Key **Certificates**).
- G1319: Ensure applications can recover data encrypted with legacy keys provided by the DoD **PKI** Key Recovery Manager (**KRM**).
- G1320: Use a minimum of 128 bits for **symmetric keys**.
- G1321: Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.
- G1322: Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of encrypting and decrypting data using the **Triple Data Encryption Algorithm** (**TDEA**).
- G1323: Generate random **symmetric encryption** keys when using symmetric encryption.
- G1324: Protect **symmetric keys** for the life of their use.
- G1325: Encrypt **symmetric keys** when not in use.
- G1326: Ensure applications are capable of producing Secure Hash Algorithm (**SHA**) **digests** of **messages** to support verification of DoD **PKI** signed objects.
- G1327: Enable an application to obtain new **Certificates** for subscribers.
- G1328: Enable an application to retrieve **Certificates** for use, including relying party operations.
- G1330: Ensure applications are capable of checking the status of **Certificates** using a **Certificate Revocation List** (**CRL**) if not able to use the **Online Certificate Status Protocol** (**OCSP**).
- G1331: Ensure applications are able to check the status of a Certificate using the **Online Certificate Status Protocol** (**OCSP**).
- G1333: Only use a **Certificate** during the Certificate's validity range, as bounded by the Certificate's "Validity - Not Before" and "Validity - Not After" date fields.
- G1335: Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.
- G1338: Applications and **Certificates** need to be able to support multiple organizational units.
- G1341: Use a security manager support to restrict application access to privileged system resources.
- G1342: Restrict direct access to class internal variables to functions or methods of the class itself.

- G1344: Encrypt sensitive data stored in configuration or resource files.
- G1357: Do not rely solely on transport level security like **SSL** or **TLS**.
- G1362: Validate incoming XML-based messages using a **schema**.
- G1363: Do not use clear text passwords.
- G1364: Hash all passwords using the combination of a timestamp, a **nonce** and the password for each **message** transmission.
- G1365: Specify an expiration value for all security tokens.
- G1366: Digitally sign all **messages** where non-repudiation is required.
- G1367: Digitally sign **message** fragments that are required not to change during transport.
- G1369: Digitally sign all requests made to a security token service.
- G1371: Use the **Digital Signature Standard** for creating **Digital Signatures**.
- G1372: Use an X.509 **Certificate** to pass a **Public Key**.
- G1373: **Encrypt messages** that cross an **IA** boundary.
- G1374: Individually **encrypt** sensitive **message** fragments intended for different intermediaries.
- G1377: Use **LDAP** 3.0 or later to perform all connections to LDAP repositories.
- G1378: Encrypt communication with **LDAP** repositories.
- G1346: Audit database access.
- G1347: Secure remote connections to a database.
- G1349: Validate all input that will be part of any dynamically generated **SQL**.
- G1350: Implement a strong password policy for **RDBMS**.
- G1351: Enhance database security by using multiple user accounts with constraints.
- G1619: Configure **clients** with a **Common Access Card** (**CAC**) reader.
- G1652: Use DoD **PKI** X.509 **certificates** for **servers**.
- G1380: Use the **XACML** 2.0 standard for **SAML**-based rule engines.
- G1797: Use a minimum of 1024 bits for **asymmetric keys**.

Best Practices

- BP1375: Use **asymmetric encryption** for **SOAP**-based **Web services**.

P1245: Design Tenet: Mediate Security Assertions

Use security assertions or security tokens to convey user authentication and access authorization to a service provider. Security assertions and tokens are statements that an entity the service provider trusts has generated and validated.

Considerations

Security Assertions

- Use an XML-based standard such as the **Security Assertion Markup Language** (**SAML**) to transfer assertions.
- For close community configurations, start with Kerberos security tokens. Establish implicit trust relationships between entities to circumvent formal validations through the use of trusted channels (e.g., **SSL** transfers).
- Transfer security tokens or security assertions using the general purpose mechanism provided for associating security tokens or assertions with SOAP message contents as specified in the WS-Security Standard. Kerberos and other tokens shall use the Binary Security Token provision. Use SAML assertions in the context of WS-Security as specified in the upcoming WS-Security SAML Token Profile. [R1246]

Chained Requests

- When requests need to be chained (i.e., forwarded to third parties), the security assertions must cover the origin and destination, all intermediate assertions, and the required chain of trust. Earlier request implementations may separate a chained request into separate transactions.

Guidance

- G1379: Use **SAML** version 2.0 for representing security assertions.
- G1380: Use the **XACML** 2.0 standard for **SAML**-based rule engines.
- G1359: Bind **SOAP Web service** security policy assertions to the service by expressing them in the associated **WSDL** file.
- G1357: Do not rely solely on transport level security like **SSL** or **TLS**.
- G1322: Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of encrypting and decrypting data using the **Triple Data Encryption Algorithm** (**TDEA**).

P1246: Design Tenet: Cross-Security-Domains Exchange

Exchange information across security boundaries using air-gap interfaces, electronically enforced one-way interfaces, content-based encryption, content-sensitive security guards, multilevel trusted databases, and multilevel systems. The data exchange may be from low to high or high to low. In an NCW environment, many of the service requests and their corresponding trust assertions may have to cross security boundaries; that is, they must originate and terminate at entities with different security classification levels.

Considerations

Cross-Domain Services

- In a net-centric environment, enterprise-wide services are the most efficient way to handle data exchange transactions and implement cross-domain solutions. Develop special cross-domain services to provide validated resources capable of transferring information between security domains operating at different security classifications. To support net-centric warfare effectively, cross-domain solutions must transition from current models to an agile and flexible, robust and available, trusted yet economical solution set. The most effective method is to provide those services at the enterprise level, compatible with the **Global Information Grid** (**GIG**) and **Net-Centric Enterprise Services** (**NCES**).
- Incorporate the capabilities and procedures of centralized cross-domain solutions as they become available. If possible, systems should demonstrate an evolution towards these enterprise-wide solutions. Rely on existing secure guard solutions or one-way solutions until enterprise-wide solutions are available.

Note: *See the following perspectives for additional considerations: Trusted Guards [P1150] and Cross-Domain Interoperation [P1169].*

Guidance

- G1341: Use a security manager support to restrict application access to privileged system resources.
- G1379: Use **SAML** version 2.0 for representing security assertions.
- G1380: Use the **XACML** 2.0 standard for **SAML**-based rule engines.
- G1613: Prepare a **Node** to host new **Component** services developed by other Nodes or by the **enterprise** itself.
- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.

Best Practices

- BP1698: Plan for the event that **Component** services within a **Node** cannot be invoked across security domains.
- BP1669: Select **XML**-capable **trusted guards**.
- BP1691: Use **Node** implemented **Service Discovery** (**SD**) to meet compartmentalization needs.
- BP1614: Prepare a **Node** for the possibility of becoming a new **Component** service within another Node.

P1247: Design Tenet: Encryption and HAIPE

Enterprise services must enable secure transmission of identification and role assertions through the use of **trusted paths**. A trusted path is a communications path where there is confidence alteration of data has not occured during transport and the data are timely.

> ***Note:*** *The definition of "timely" is not the same for all types of information systems. Services should specify an appropriate definition based on the type of information system (e.g., event-driven, transaction-based) and the type of security threat (e.g., replay attack).*

Considerations

- Use **Secure Scokets Layer** (**SSL**), Internet Protocol Security (IPSec), or **High Assurance Internet Protocol Encryption** (**HAIPE**) protocols to secure transmission of identification and role assertions in a **TCP/IP** environment. Incorporating message-level encryption may provide additional security.

Guidance

- G1305: Ensure the separation of **encrypted** and unencrypted information.
- G1322: Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of encrypting and decrypting data using the **Triple Data Encryption Algorithm** (**TDEA**).
- G1323: Generate random **symmetric encryption** keys when using symmetric encryption.
- G1324: Protect **symmetric keys** for the life of their use.
- G1325: Encrypt **symmetric keys** when not in use.
- G1344: Encrypt sensitive data stored in configuration or resource files.
- G1357: Do not rely solely on transport level security like **SSL** or **TLS**.
- G1363: Do not use clear text passwords.
- G1364: Hash all passwords using the combination of a timestamp, a **nonce** and the password for each **message** transmission.
- G1366: Digitally sign all **messages** where non-repudiation is required.
- G1367: Digitally sign **message** fragments that are required not to change during transport.
- G1369: Digitally sign all requests made to a security token service.
- G1371: Use the **Digital Signature Standard** for creating **Digital Signatures**.
- G1372: Use an X.509 **Certificate** to pass a **Public Key**.
- G1373: **Encrypt messages** that cross an **IA** boundary.
- G1374: Individually **encrypt** sensitive **message** fragments intended for different intermediaries.
- G1376: Do not **encrypt** key elements that are needed for correct **SOAP** processing.

- G1378: Encrypt communication with **LDAP** repositories.
- G1381: Encrypt all sensitive persistent data.
- G1320: Use a minimum of 128 bits for **symmetric keys**.
- G1326: Ensure applications are capable of producing Secure Hash Algorithm (**SHA**) **digests** of **messages** to support verification of DoD **PKI** signed objects.
- G1321: Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.
- G1607: Configure routers according to **National Security Agency** (NSA) Router Security Configuration guidance.
- G1797: Use a minimum of 1024 bits for **asymmetric keys**.

Best Practices

- BP1375: Use **asymmetric encryption** for **SOAP**-based **Web services**.

P1248: Design Tenet: Employment of Wireless Technologies

Considerations

- All data transmissions need integrity assurances that the information has not been altered. For transmission of sensitive or classified information, there should also be assurances that the information has not been exposed to unauthorized users. In the case of wireless technologies, consider those assurances in the context of lack of finite boundaries for information protection, and the possibilities of spoofing (i.e., unauthorized insertions of information). Many standards are being developed for the protection of wireless networks using cryptographic means.

- Systems should encrypt all traffic when using wireless technologies using established standards.

Best Practices

- BP1880: Justify, document, and obtain a waiver for all radio terminal acquisitions that are not JTRS/SCA compliant.

P1251: Other Design Tenets

Provide boundary or perimeter protection for **service-oriented architectures** (SOAs) to help prevent penetration from non-DoD external sources. The main defense security regulations, namely DoD 8500 Series and DCID 6/3 [R1247] , apply to **SOA** components. Some of the regulations may not directly apply, or they may require special considerations when applied to SOAs.

Considerations

Integrity and Confidentiality

- Encrypt requests and responses to achieve the appropriate level of confidentiality protection using protocols such as the following:
    - Secure Socket Layer (SSL) or Transport Level Security (TLS) for transport layer security
    - Internet Protocol Security (IPSec) for network layer
    - Secure Multi-purpose Internet Mail Extensions (S/MIME) for email traffic
- Migrate toward message-level encryption using standards such as XML-Encryption and provide message integrity protection using standards such as XML-Digital Signature.
- Include timestamps within messages to prevent recording and playback of messages. All timestamps must use Coordinated Universal Time (UTC), also referred to as Greenwich Mean Time (GMT) or Zulu (Z) time.

Firewall Configurations

- Continue using firewalls and proxy servers to protect the physical boundary of clusters of equipment supporting SOAs. Firewalls must prevent unauthorized penetrations; they require carefull programming to reduce the inherent additional risks of SOAs.
- An example of one such risk would be allowing inbound HTTP/HTTPS access to Web-based applications. This may allow an ill-intended SOAP message to cause an internal application buffer overflow while looking completely benign to the firewall. To help prevent such a threat, use XML-capable firewalls as they become available.

Intrusion Detection Systems

- Use adequate monitoring to determine anomalies or failures that can impair mission performance. Intrusion detection systems should detect unauthorized access and penetration attempts. Use detection and protection mechanisms to detect and prevent illicit actions automatically, and complement them with manual reporting of anomalies or specially detected events. Enable automatic reconfiguration or recovery features only for limited and well-defined conditions.

Intrusion Reporting

- A service-oriented architecture requires some centralization of automated reports which, when coupled with correlation and analysis of events detected at multiple nodes, helps establish enterprise security awareness. The scope of the environment conducting the correlation depends on the availability of software agents in individual nodes and the availability of resources that can establish the correlation of events. The scope may range from a few systems at a given location to all activities within a theater of operations. An even broader analysis may occur through manual reporting at an enterprise-wide level.

Audit Events Linkage

- Configure and use individual system audit mechanisms. For SOAs, complement audits with mechanisms that correlate events in different nodes and provide network-wide forensics. Time stamping and logging of all inter-node messages help link events and actions involving multiple nodes. Use UTC for time stamping.

Use of Audits for Attribution

- Use logging and request auditing to satisfy attribution requirements (i.e., determination of the individual responsible for the action). This should occur at both the requestor and service provider sites.

GIG Policy Compliance

- Develop systems in accordance with the IA requirements in DoD Instruction 8500.2 [R1198] for the appropriate Mission Assurance Category and Sensitivity Level. Systems dealing with intelligence sources and methods must also comply with DCID 6/3. [R1247] Also leverage the guidance and technologies described in DoD CIO Guidance and Policy Memorandum 6-8510, ***DoD GIG Information Assurance*** [R1251] and the ***End-to-End Information Assurance of the GIG***. [R1252]

Certification and Accreditation

- Certify and accredit all systems in accordance with DoD Instruction 8510.01, ***DoD Information Assurance Certification and Accreditation Process (DIACAP)***. [R1291] In addition, Air Force systems should comply with the certification and accreditation section in Air Force Instruction 33-202, ***Network and Computing Security***. [R1249]

Guidance

- G1301: Practice layered security.
- G1302: Validate all inputs.
- G1305: Ensure the separation of **encrypted** and unencrypted information.
- G1359: Bind **SOAP Web service** security policy assertions to the service by expressing them in the associated **WSDL** file.
- G1363: Do not use clear text passwords.
- G1364: Hash all passwords using the combination of a timestamp, a **nonce** and the password for each **message** transmission.
- G1365: Specify an expiration value for all security tokens.
- G1369: Digitally sign all requests made to a security token service.
- G1372: Use an X.509 **Certificate** to pass a **Public Key**.
- G1376: Do not **encrypt** key elements that are needed for correct **SOAP** processing.
- G1339: Practice defensive programming by checking all method arguments.
- G1340: Log all exceptional conditions.
- G1346: Audit database access.
- G1348: Log database **transactions**.
- G1349: Validate all input that will be part of any dynamically generated **SQL**.

- G1622: Implement **commercial off-the-shelf** (COTS) software that protects against malicious code on each operating system in the Node in accordance with the Desktop Application **Security Technical Implementation Guide** (STIG).

- G1623: Implement personal **firewall** software on **client** or **server** hardware used for remote connectivity in accordance with the Desktop Applications, Network and Enclave **Security Technical Implementation Guides** (**STIGs**).

- G1624: Install anti-**spyware** on all **client** and **server** hardware.

- G1632: Certify and accredit Nodes with all applicable DoD **Information Assurance** (IA) processes.

- G1633: Host only DoD **Information Assurance** (IA) certified and accredited **Components**.

- G1634: Certify and accredit **Components** with all applicable DoD **Information Assurance** (IA) processes.

- G1662: Follow the guidance provided in the **Security Technical Implementation Guide** (STIG) for **Domain Name System** (DNS) implementations.

- G1667: Implement **Virtual Private Networks** (VPNs) in accordance with the guidance provided in the Network **Security Technical Implementation Guide** (STIG).

P1241: Transport

The Transport Infrastructure is a foundation for net-centric transformation in DoD. To realize the vision of the **Global Information Grid** (**GIG**), the Assistant Secretary of Defense for Networks and Information Integration/DoD Chief Information Officer (ASD(NII)/DoD CIO) has called for a dependable, reliable, and ubiquitous network that eliminates stovepipes and responds to the dynamics of the operational scenario. To construct the Transport Infrastructure, DoD will do the following:

- Follow the **Internet** model
- Create the GIG from smaller component building blocks
- Design with interoperability, flexibility to evolve, and simplicity in mind
- Provide a common, black-core IP network for both unclassified and encrypted classified information

Both users and providers of transport servivces must conform to established and evolving transport-related standards and guidelines. The DoD IT Standards Registry (DISR) [R1179] is the primary source for DoD-adopted standards.

Note: *See the Node Transport perspective [P1148] for further guidance.*

Detailed Perspectives

- Design Tenet: IPv6
- Design Tenet: Packet Switched Infrastructure
- Design Tenet: Layering and Modularity
- Design Tenet: Transport Goal
- Design Tenet: Network Connectivity
- Design Tenet: Concurrent Transport of Information Flows
- Design Tenet: Differentiated Management of Quality-of-Service
- Design Tenet: Inter-Network Connectivity
- Design Tenet: DoD IT Standards Registry (DISR)
- Design Tenet: RF Acquisition
- Design Tenet: Joint Net-Centric Capabilities
- Design Tenet: Operations and Management of Transport and Services

P1255: Design Tenet: IPv6

In the next few years, the adoption of **IPv6** throughout the DoD and other Federal Agencies will pass a major implementation threshold. Most DoD bases and other facilities will be IPv6 capable. Most of the key components of the technology are in place for native deployment of IPv6 or dual existence of **IPv4** and IPv6.

A 9 June 2003 ASD(NII)/DoD CIO memo, ***Internet Protocol Version 6 (IPv6)*** is the first in a series of memos addressing DoD transition to IPv6 [R1190] . The main points of the directives follow:

- The tentative original goal for IPv6 transition completion is FY08.
- DoD is conducting enterprise-wide deployment of IPv6 in a controlled, integrated and cohesive manner (see the DoD IPv6 Transition Plan [R1254]).
- The DoD IPv6 Transition Office established within **DISA** is responsible for coordinating transition efforts, providing required infrastructure, and insuring that unified solutions are used across DoD. Each Service has a Transition Office responsible for providing technical guidance and transition governance to programs. This includes developing transition plans (subject to coordination into a master plan by DISA), dispensing IP addresses originating from DISA, implementing waiver policy, etc.
- A mandate, to minimize costs of transition, is that all **GIG** assets being developed, procured or acquired must be IPv6 capable (in addition to maintaining interoperability with IPv4 capabilities). The DoD CIO directives contain an outline for the "IPv6 capable" requirement, while a detailed specification is still under development.
- The transition to IPv6 should be accomplished through the normal technical refresh cycle whenever possible.

Considerations

Support IPv6 Transition

- Be able to interoperate with interfacing transport service providers who use either IPv6 or IPv4 during the transition from IPv4. New applications should be IP version agnostic and shall employ an operating system that supports both IPv4 and IPv6. For existing IPv4 service users, the governing authority (e.g., Component IPv6 Transition Office) should develop and approve IPv6 migration plans.
- Transport service providers interfacing with non-transitioned networks must support both IPv6 and IPv4 during the transition from IPv4. Mechanisms proposed to allow the two protocols to coexist and inter-operate during the transition phase from IPv4 to IPv6 include the following:
    - Incorporating both IPv4 and IPv6 support in routers and computers; this is called ***dual stacking***. This is a preferred way to ensure the interoperability between systems during the transition period.
    - Transporting IPv6 traffic through IPv4 networks by encapsulating IPv6 packet in IPv4 and vice-versa; this is called ***tunneling***. During the initial enabling of IPv6 in operational environments in controlled enclaves, tunneling becomes a useful communication mechanism between the enclaves. Tunneling should be considered only as a temporary solution.
    - Placing translation gateways between IPv4 and IPv6 networks or hosts. This is the only mechanism allowing a native IPv4-only device to communicate with IPv6-only device. The expectation is that these devices will not be needed until the later stages of transition for dominant IPv6 devices to communicate with some lingering native IPv4 legacy devices [R1255] .
- In all cases, IPv6 transport provider planning must be coordinated with the Service IPv6 Transition Office.

Support IPv6 IP security features for data integrity and confidentiality.

- IPv6 provides improved security features in comparison to IPv4 through IPSec and mandatory support for end-to-end security. The Service Transition Office should be able to provide guidance on utilizing any of the IPv6 security features in the context of the service enterprise transition plan.

- Implement DoD-adopted IPv6 standards and products. The list of standards directly relevant to DoD and approved for the use on DoD networks is maintained in the DISR [R1179] .

Guidance

- G1586: Provide a transport infrastructure for the Node that is **Internet Protocol Version 6** (IPv6) capable in accordance with the appropriate governing transition plan.

- G1587: Prepare an **Internet Protocol Version 6** (IPv6) transition plan for the Node.

- G1588: Coordinate an **Internet Protocol Version 6** (IPv6) transition plan for a Node with the **Components** that comprise the Node.

- G1589: Address issues in the appropriate governing **IPv6** transition plan as part of the Internet Protocol Version 6 (IPv6) Transition Plan for a Node.

- G1590: Include transition of all the impacted elements of the network as part of the **Internet Protocol Version 6** (IPv6) Transition Plan for a Node.

- G1591: Prepare **IPv6** Working Group products as part of the Internet Protocol Version 6 (IPv6) transition plan for a Node.

- G1592: Include interoperability testing in the plan as part of the **Internet Protocol Version 6** (IPv6) transition plan for a Node.

- G1599: Support both **Internet Protocol Version 4** (IPv4) and **Internet Protocol Version 6** (IPv6) simultaneously in the Node's **Domain Name System** (DNS) service.

- G1600: Obtain from DISA any and all **Internet Protocol Version 6** (IPv6) addresses used on DoD systems in the Node.

- G1595: Implement **Domain Name System** (DNS) to manage hostname/address resolution within the Node.

Best Practices

- BP1863: Make shareable data assets visible, even if they are not accessible.

- BP1870: Conform to DoD-specified data publication methods that are consistent with **Global Information Grid** (**GIG**) enterprise and user technologies per DoD Directive 8101.1. [R1166]

- BP1705: Design **DNS** infrastructure in accordance with appropriate governing **IPv6** Transition Office requirements.

- BP1663: Design a **Domain Name System** (DNS) in coordination with the appropriate governing Internet Protocol Version 6 (IPv6) Transformation Office.

P1260: Design Tenet: Packet Switched Infrastructure

The **Global Information Grid** (**GIG**) includes a number of component networks. Each must pass data both internally among its network members and externally to or from other GIG component systems. As such, the design of the Internet model that applies to the development of the GIG transport infrastructure needs to be an IP datagram delivery system. The delivery system consists of a packet-switched communications facility in which a number of distinguishable component networks (including any networks external to this system) are connected together using routers. Technologies such as routing standards and quality of service (QoS) mechanisms are needed to achieve the end-to-end functionality the GIG requires. Design and apply these within the framework of packet-switched transport infrastructure.

Considerations

- Implement interface(s) to one and only one network layer protocol (Layer-3 in the ***OSI Reference Model***) for datagrams. This applies to transport service providers and consumers and to datagrams passed within a component network and those destined for external networks. The fundamental goal is a single inter-network protocol.

- GIG component system designers should consider how the component transport infrastructure will accept externally-generated IP datagrams that are destined for hosts inside their system. This allows their system to "attach" to the GIG. The designers should also consider how their component infrastructure will deliver internally generated IP datagrams to hosts outside their system, and how it will serve as a transit network for externally generated IP datagrams.

Guidance

- G1595: Implement **Domain Name System** (DNS) to manage hostname/address resolution within the Node.

- G1596: Use **Domain Name System** (DNS) **Mail eXchange (MX) Record** capabilities to configure electronic mail delivery to the Node.

- G1598: Allow dynamic **Domain Name System** (DNS) updates to the Node's internal DNS service by local **Dynamic Host Configuration Protocol** (DHCP) **server(s)**.

- G1601: Use configurable **routers** to provide dynamic **Internet Protocol** (IP) address management using **Dynamic Host Configuration Protocol** (DHCP).

- G1602: Use configurable **routers** to provide static **Internet Protocol** (IP) addresses.

- G1604: Use configurable **routers** to provide time synchronization services using **Network Time Protocol** (NTP).

- G1605: Use configurable **routers** to provide **multicast** addressing.

- G1606: Manage **routers** remotely from within the **Node**.

- G1607: Configure routers according to **National Security Agency** (NSA) Router Security Configuration guidance.

- G1608: Obtain the reference time for the Node time service from a globally synchronized time source.

- G1609: Arrange for a backup time source for the Node time service.

- G1610: Configure the **Dynamic Host Configuration Protocol** (DHCP) services to assign **multicast** addresses.

- G1611: Implement Internet Protocol (**IP**) gateways to interoperate with the **Global Information Grid** (GIG) until IP is supported natively for **Components** that are not IP networked, such as aircraft data links (**Link-16**, **SADL**, etc.).

- G1612: Implement Internet Protocol (**IP**) gateways as a **service**.

Best Practices

- BP1864: Layer architectures to support clear boundaries between data management, presentation, and business logic functionality.
- BP1877: Align end-to-end interoperable management of **QoS** with external networks.
- BP1878: Quantitative measures of QoS requirements should be supportable.
- BP1879: The program, project or initiative should align with the DoD Qos/CoS Working Group Roadmap.
- BP1874: Develop methods to forward IP datagrams from external networks.
- BP1876: Provide a priority-based differentiated management of **quality-of-service** for traffic based on class of user, application, or mission.

P1261: Design Tenet: Layering and Modularity

Change is probably the only inviolable characteristic of the commercial **Internet** model. Moreover, change occurs at different rates in different elements of the network/protocol stack. Design the **Global Information Grid** (**GIG**) transport infrastructure to accommodate that change. The most effective way to allow differential change in a system is through modular, layered design.

Although market forces and commercial practice sometimes have deprecated the International Organization for Standardization (ISO) Open System Interconnection (OSI) Model, it still provides excellent guidelines for implementing a layered design. These guidelines still apply to the development of the GIG transport infrastructure.

In a layered design, each layer is independent and adds value to the set of services offered by lower layers. The services provided to and from a layer are well defined; however, the precise approach for providing these services is not specified. ISO defined a number of principles to consider when developing a layered design and applied those principles to develop the seven-layer OSI Model.

While a seven-layer approach may not be the solution for the GIG transport infrastructure, GIG component system designers should consider the principles ISO defined to facilitate interoperability and to reduce technology interdependencies that add to system complexity. The following considerations include a subset of these principles that apply to the GIG transport infrastructure.

Considerations

Define Layer Boundaries and Interfaces

- Implement one or more interfaces to the defined transport service delivery point(s) or interface boundaries, where the services description can minimize the number of interactions across the interface boundary(ies). The networks should provide the interface boundary definition(s). To the maximum extent possible, functionality implemented within each OSI layer of the transport service implementation should only interface with the adjacent lower layer via defined interfaces. The goal is to minimize the cross-layer physical and functional interdependencies to facilitate GIG transport infrastructure growth and interoperability.

Ensure Functions are Modular and Separable

- Create a layer of easily localized functions. These functions should enable developers to totally redesign the layer and its protocols to take advantage of new advances in architectural, hardware, or software technology without changing the services and interfaces with the adjacent layers.
- Identify all instances in the transport infrastructure where a logical or physical coupling or dependency exists between different layers of the protocol stack. The goal is to minimize the cross-layer physical and functional interdependencies to facilitate GIG transport infrastructure growth and interoperability.

Minimize Complexity of Layered Implementation

- Keep the number of layers within networks small enough to reduce the complexity of describing, integrating, and maintaining the layers.

Guidance

- G1301: Practice layered security.

Best Practices

- BP1876: Provide a priority-based differentiated management of **quality-of-service** for traffic based on class of user, application, or mission.

- BP1790: Stipulate that the Offeror is to describe how the proposed technical solution reuses services from other systems or demonstrates composeability and extensibility by building from existing reusable components and/or services.

- BP1829: Use the **Data Distribution Service** (DDS) **`OWNERSHIP`** **Quality of Service** (QoS) kind set to **`EXCLUSIVE`** when multiple **DataWriters** cannot write each unique data-object within a DDS **Topic** simultaneously.

P1262: Design Tenet: Transport Goal

A design goal of the **Global Information Grid** (**GIG**) is network convergence with voice, video, and other multimedia traffic packetized and transported along with data traffic over a common **Internet Protocol** (**IP**) network. Another transport goal is the convergence of encrypted classified information flows on a common black IP network. This corresponds to the direction of commercial industry, where telecommunications providers and corporate telephony are migrating to IP.
A primary benefit of convergence is that it eliminates the expensive hardware and complexity of separate, dedicated networks that support serial-based traffic (e.g., voice and video teleconferencing). Other benefits include greater efficiency of bandwidth and the ability to introduce new features based on converged services.

Considerations

Support Interfaces with Converged Traffic Networks

- Implement interfaces to, or transition to, a transport infrastructure supporting full convergence of traffic on a single IP inter-network, using DoD-adopted standards and DISA/**JITC**-certified (voice) solution sets.
- Identify and minimize all instances where performance standards cannot be met using a converged transport infrastructure (e.g., where dedicated, single-traffic-type transport service is required). The goal is to minimize cross-layer physical and functional interdependencies to facilitate GIG transport infrastructure growth and interoperability.
- Voice, video, and other multimedia traffic have relatively strict delivery requirements with regard to latency and jitter. This requires networks to support the QoS features identified in the Design Tenet: Differentiated Management of Quality-of-Service.
- The DoD-adopted set of standards appears in the DoD IT Standards Registry (DISR) [R1179] . DISR specifies standards for Voice over IP (VoIP) and video teleconferencing (VTC) based on the **International Telecommunication Union** (**ITU**) standard H.323.
- Voice over IP (VoIP) refers to a set of standards and technologies that allow transmission of voice data over IP networks. The industry has embraced two different sets of standards:
    - ITU H.323 is the more mature and complete set of standards, which encapsulates Integrated Services Digital Network (ISDN) call signaling over an IP-based network.
    - A more recent set of standards, developed by the **Internet Engineering Task Force** (**IETF**), is based on the Session Initiation Protocol (SIP). The SIP standard concerns simple call placement and is designed to be easily expandable.
- Since there are currently two options for VoIP, the DoD plans to select a set of mandated standards within the DISR.
- Video teleconferencing over IP is based on ITU H.323. This is an umbrella standard of ITU recommendations that address audio, video, signaling, and control for packet-switched networks.

Guidance

- G1585: Provide a transport infrastructure for the Node that implements **Global Information Grid** (GIG) **Information Assurance** (IA) boundary protections.
- G1584: Provide a transport infrastructure that is shared among **Components** within the Node.
- G1586: Provide a transport infrastructure for the Node that is **Internet Protocol Version 6** (IPv6) capable in accordance with the appropriate governing transition plan.

Best Practices

- BP1864: Layer architectures to support clear boundaries between data management, presentation, and business logic functionality.

- BP1875: Describe the process and protocols used to provide concurrent traffic from multiple security domains on a single **IP** internetwork.

- BP1877: Align end-to-end interoperable management of **QoS** with external networks.

- BP1878: Quantitative measures of QoS requirements should be supportable.

- BP1879: The program, project or initiative should align with the DoD Qos/CoS Working Group Roadmap.

- BP1594: Examine the use of **Transmission Control Protocol** (TCP) extentions and other transport protocols that have been designed to mitigate risk for high bandwidth, high latency satellite communications.

- BP1876: Provide a priority-based differentiated management of **quality-of-service** for traffic based on class of user, application, or mission.

P1263: Design Tenet: Network Connectivity

Provide network connectivity to all end points, such as wide- and local-area networks, and direct connections to mobile end users. This perspective addresses the Open System Interconnection (OSI) Model Layer-2 or terminal-to-network interfaces.

Considerations

Manage Scalability and Complexity

- Quantitatively evaluate scalability before formulating a final design. The evaluation should identify any transport infrastructure design drivers regarding the number of hosts that need to be supported and/or number of networks that are required to support the technologies chosen for the specific transport service or infrastructure use.

- One way to reduce complexity is to use a minimal set of standards/protocols in developing the **Global Information Grid** (**GIG**) transport infrastructure. This implies that any selected standard/protocol has the capacity to serve as large a percentage of the GIG as possible. Component systems of the GIG should select standards/protocols that can scale to the enterprise. GIG component system designers should evaluate their transport infrastructure design to identify any instances where different technology/protocols perform the same function (e.g., internal routing).

Optimize Use of COTS Products

- Use open, **commercial-off-the-shelf** (**COTS**) products as much as possible. Government-off-the-shelf (GOTS) and/or vendor-unique products may lead to interoperability and evolvability issues. Use them only when there is an overarching, unique, DoD requirement driving that selection.

- Document the justification for the use of any protocols, standards, etc., that are not included the DoD IT Standards Registry and/or could not be purchased off-the-shelf from a commercial networking vendor.

Guidance

- G1605: Use configurable **routers** to provide **multicast** addressing.
- G1606: Manage **routers** remotely from within the **Node**.
- G1602: Use configurable **routers** to provide static **Internet Protocol** (IP) addresses.
- G1601: Use configurable **routers** to provide dynamic **Internet Protocol** (IP) address management using **Dynamic Host Configuration Protocol** (DHCP).
- G1604: Use configurable **routers** to provide time synchronization services using **Network Time Protocol** (NTP).
- G1609: Arrange for a backup time source for the Node time service.
- G1607: Configure routers according to **National Security Agency** (NSA) Router Security Configuration guidance.
- G1330: Ensure applications are capable of checking the status of **Certificates** using a **Certificate Revocation List** (**CRL**) if not able to use the **Online Certificate Status Protocol** (**OCSP**).
- G1610: Configure the **Dynamic Host Configuration Protocol** (DHCP) services to assign **multicast** addresses.
- G1608: Obtain the reference time for the Node time service from a globally synchronized time source.

- G1582: In Node **Enterprise Service** schedules, include version numbers of standard Enterprise Services interfaces being implemented.

Best Practices

- BP1830: Use the **Data Distribution Service** (DDS) Content Profile to tailor subscription message data.
- BP1651: Do not implement **server** side **CES** functionality for **Components**.
- BP1845: Consider key enterprise-level concerns when planning and executing a migration to net-centricity and SOA.

P1264: Design Tenet: Concurrent Transport of Information Flows

This tenet addresses the use of Inline Network Encryptors (INEs) that allow all security domains to be "known" globally to the Open System Interconnection (OSI) Model Layer-3 encrypted backbone network. This is a fundamental shift from current link-by-link encryption. Utilizing a Black Core network should provide a significantly streamlined communications infrastructure that also makes more efficient use of the available bandwidth through the invocation ofquality-of-service/ class-of-service (QoS/CoS) based IP datagram multiplexing.

High Assurance Internet Protocol Encryptor (**HAIPE**) devices are among the critical technologies that should enable the Black Core IP-network vision to become a reality. However, a number of technical challenges must be solved before the vision can be realized across all functional domains and **Communities of Interest** (**COIs**). These include the following:

- Support for IP-based QoS/CoS
- Support for dynamic unicast IP routing
- Support for dynamic multicast IP routing
- Support for mobility
- Support for simultaneous IPv6 and IPv4 operation

Considerations

Implement INE Standards and Products to Support Traffic Convergence

- Government-off-the-Shelf (GOTS) and/or vendor-unique products may lead to interoperability and evolvability issues. Use them only when there is an overarching, unique, DoD requirement driving that selection.
- Implement DoD-adopted INE standards and products, when available, to support traffic convergence from multiple security domains on a single IP inter-network. Currently, DoD is engaged in **IETF**-standards working groups and vendor communities to accelerate development of new standards in the areas of security, tactical communications, QoS, and reliable networking. Some standards have been adopted for QoS and HAIPE. A product list is in development for infrastructure, hardware, software, and other categories of IPv6 products.

Document Approach to Information Infrastructure with Black Core

- GOTS and/or vendor-unique products may lead to interoperability and evolvability issues. Use them only when there is an overarching, unique, DoD requirement driving that selection.
- Document the approach to providing an information infrastructure with a Black Core.

Guidance

- G1607: Configure routers according to **National Security Agency** (NSA) Router Security Configuration guidance.

Best Practices

- BP1875: Describe the process and protocols used to provide concurrent traffic from multiple security domains on a single **IP** internetwork.
- BP1879: The program, project or initiative should align with the DoD Qos/CoS Working Group Roadmap.

- BP1880: Justify, document, and obtain a waiver for all radio terminal acquisitions that are not JTRS/SCA compliant.
- BP1670: Monitor Black Core implementation issues and prepare a plan for local implementation in coordination with system programs fielded within the Node.
- BP1671: Consider Black Core transition whenever there is a significant Node network design or configuration decision to make in an effort to avoid costly downstream changes caused by Black Core transition.

P1265: Design Tenet: Differentiated Management of Quality-of-Service

Some applications in the **Global Information Grid** (**GIG**) require firm service guarantees, while others operate correctly if they receive services that are differentiated with respect to one or more performance characteristics.

Differentiated Services or DiffServ aggregates flows into coarse classes and then treats the packets in these classes differentially. Due to this aggregation, and the resulting absence of a need to consider individual flows beyond the edges of an internet, DiffServ exhibits good scaling properties. However, in the absence of additional mechanisms, DiffServ provides only preferential, differentiated levels of service and not guarantees.

Considerations

Support Quality of Service (QoS) and Class of Service (CoS)

- Interoperate with interfacing transport service providers who use standardized DoD QoS/CoS in accordance with the DoD QoS/CoS Roadmap. As the interfacing networks are transitioned to standardized QoS/CoS, plan to migrate to maintain interoperability.
- Prioritize traffic based on class of user, application, or mission. Lower priority data flows should be preempted if a higher priority flow is initiated and insufficient resources exist to carry both flows simultaneously. This capability, referred to as Class of Service (CoS) support, corresponds approximately to the notion of Multi-Level Priority and Preemption (MLPP). The GIG and its components should support both QoS and CoS in accordance with the DoD QoS/CoS Roadmap and policies

Guidance

- G1771: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of a **publisher**.
- G1801: Explicitly define a **Topic Quality of Service** (QoS) for each **Data Distribution Service** (DDS) Topic within a DDS **Domain**.
- G1803: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe real-time messaging criteria for **Publishers**.
- G1804: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe **DataWriter**.
- G1805: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of the **Subscriber**.
- G1806: Explicitly define the Request-Offered **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of the **DataReader**.
- G1808: Handle all **Data Distribution Service** (DDS) **Quality of Service** (QoS) contract violations using one of the **Subscriber access APIs**.

Best Practices

- BP1876: Provide a priority-based differentiated management of **quality-of-service** for traffic based on class of user, application, or mission.
- BP1877: Align end-to-end interoperable management of **QoS** with external networks.

- BP1878: Quantitative measures of QoS requirements should be supportable.
- BP1879: The program, project or initiative should align with the DoD Qos/CoS Working Group Roadmap.

P1266: Design Tenet: Inter-Network Connectivity

A fundamental tenet of the commercial Internet model is that the complexity of the Internet belongs at the edges. Certain required end-to-end functions can only be performed correctly by the end systems themselves. Any network, however carefully designed, will be subject to failures of transmission at some statistically determined rate.

The best way to cope with this is to accept it and give responsibility for the integrity of communication to the end systems. This principle drives the complexity of the network to the edge and limits state information held inside the network. This increases the robustness of end-to-end communications since application state can now only be destroyed by a failure of the end systems.

Many issues need to be resolved to mature the guidance for this tenet, especially for transport users whose data traverse different media with different performance characteristics. In some situations it may not be desirable to follow this design tenet.

For example, the use of TCP proxies, which may be required to achieve adequate performance across satellite assets, runs counter to this tenet. The proxy (part of the network and not an end system) maintains state information on the TCP **session** between two end-user systems, but it cannot guarantee that the function that TCP is performing is being accomplished.

Avoid implementing "intelligence" within the network whenever possible.

Considerations

Support Inter-network Connectivity Using DoD-Adopted Standards

- Support inter-network connectivity using DoD-adopted standard protocols contained in the ***DoD IT Standards Registry (DISR)*** [R1179] , such as BGP4. Any protocols or standards that are not included in the DISR, such as performance-enhancing proxies, should be documented and justified against the resulting impact to GIG component system interoperability.

Guidance

- G1601: Use configurable **routers** to provide dynamic **Internet Protocol** (IP) address management using **Dynamic Host Configuration Protocol** (DHCP).
- G1602: Use configurable **routers** to provide static **Internet Protocol** (IP) addresses.
- G1604: Use configurable **routers** to provide time synchronization services using **Network Time Protocol** (NTP).
- G1605: Use configurable **routers** to provide **multicast** addressing.
- G1606: Manage **routers** remotely from within the **Node**.
- G1607: Configure routers according to **National Security Agency** (NSA) Router Security Configuration guidance.
- G1608: Obtain the reference time for the Node time service from a globally synchronized time source.
- G1609: Arrange for a backup time source for the Node time service.
- G1610: Configure the **Dynamic Host Configuration Protocol** (DHCP) services to assign **multicast** addresses.
- G1623: Implement personal **firewall** software on **client** or **server** hardware used for remote connectivity in accordance with the Desktop Applications, Network and Enclave **Security Technical Implementation Guides** (**STIGs**).

P1267: Design Tenet: Joint Technical Architecture [now DISR]

Note: *This topic is "Design Tenet: Joint Technical Architecture" in the Net-Centric Checklist v2.1.3 of 12 May 2004. The DISR Baseline Release 04-2.0 of 22 December 2004 replaced the JTA so this perspective refers to the DISR rather than the JTA.*

DoD-approved standards and protocols related to net-centricity are in the ***DoD Information Technology (IT) Standards Registry (DISR)***. [R1179] Programs, projects or initiatives should support computing infrastructure that is compliant with the net-centric interoperability standards in the DISR. NESI provides implementation guidance and best practices for DoD sanctioned standards and protocols. However, other standards are often useful and when a program (or project or initiative) uses them, the program manager needs to be able to justify this use. Many of the technologies and implementation specifics associated with the ASD(NII) Net-Centric Checklist Tenets are still in development and have not yet reached maturity.

Considerations

- Justify and document all standards that are not included in the DoD Information Technology (IT) Standards Registry (DISR), [R1179] especially those that impact transport service infrastructure design.

Best Practices

- BP1712: Register developed mappings in the **DoD Metadata Registry**.
- BP1875: Describe the process and protocols used to provide concurrent traffic from multiple security domains on a single **IP** internetwork.

P1269: Design Tenet: RF Acquisition

Considerations

JTRS/SCA Compliance

- Justify, document, and obtain a waiver for all radio terminal acquisitions that are not **Joint Tactical Radio System** (**JTRS**) /**Software Communications Architecture** (**SCA**) compliant and coordinate with the Office of the Secretary of Defense (OSD) and the JTRS Joint Program Executive Office (JPEO); the following references apply: [R1240] and [R1241] .

Minimize RF Bandwidth Requirements

- Use appropriate transmit protocols, compression standards, and other techniques when interfacing radio frequency (RF) networks to the **Global Information Grid** (**GIG**) environment. The RF environment, with its much more constrained and error prone propagation environment, requires techniques that minimize bandwidth requirements.

Guidance

- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.

- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.

Best Practices

- BP1715: Design SCA log services according to the OMG Lightweight Log Service Specification.

P1274: Design Tenet: Joint Net-Centric Capabilities

The Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer (ASD[NII]/DoD CIO) issued a 15 July 2003 memorandum, ***Joint Net-Centric Capabilities***, that identifies a number of key **C4ISR** programs for integrating into the **Global Information Grid** (**GIG**):

- All Space Terminal acquisitions
- All Intelligence, Surveillance, and Reconnaissance (ISR) programs
- Teleport
- Warfighter Information Network-Tactical (WIN-T)
- All radio and data link applications
- Global Command and Control System (GCCS, Joint and Service variants)
- Crypto Modernization
- Distributed Common Ground Systems (DCGS)
- All C2 programs
- Deployable Joint Command and Control (DJC2)
- High Assurance Internet Protocol Encryption (HAIPE)
- Future Combat Systems (FCS)
- Programs under the FORCEnet umbrella

The memo highlights programs that are required to develop transition plans for integrating transport components with the following GIG joint net-centric capabilities:

- Internet Protocol version 6 (IPv6)
- Net-Centric Enterprise Services (NCES)
- **Joint Tactical Radio System** (**JTRS**)/**Software Communications Architecture** (**SCA**)
- Global Information Grid Bandwidth Expansion (GIG-BE)
- Transformational Communications Satellite/Advanced Wideband System
- End-to-end information assurance

The ASD(NII) Net-Centric Checklist [R1177] also highlights the need for the programs to include in transition plans the use of guard technologies, and standards and protocols for connectivity with allied and coalition partners.

Considerations

Employ NCOW RM

- Use the Net-Centric Operations and Warfare Reference Model (NCOW RM) [R1176] to guide implementation of Joint net-centric capabilities. The reference model provides context for the types of architectures and computing infrastructures that the GIG transport systems and management functions must support.

- Use the NCOW RM to define the architectures of Joint net-centric capabilities. The GIG NetOps Architecture from GIG Version 1.0 was a central component used to develop NCOW RM. The reference model provides context for the types of architectures and computing infrastructures that the GIG transport systems and management functions must support.

Guidance

- G1629: Identify which **Net-Centric Enterprise Services** (NCES) capabilities the Node requires during deployment.

- G1576: Provide an environment to support the development, build, integration, and test of net-centric capabilities.

Best Practices

- BP1866: Coordinate with end users to develop interoperable materiel in support of high-value mission capability.

- BP1880: Justify, document, and obtain a waiver for all radio terminal acquisitions that are not JTRS/SCA compliant.

- BP1681: Make **Component** services metrics visible and accessible as part of the service registration and updated periodically.

- BP1840: Identify opportunities to apply the principles of net-centricity and SOA throughout the course of the program.

- BP1661: Engage with the **Net-Centric Enterprise Services** (NCES) program office to explore approaches for mobile use of the **Core Enterprise Services** (CES) services in mobile Nodes that rely on **Transmission Control Protocol/Internet Protocol** (TCP/IP) for inter-node communication.

- BP1837: Update the **net-centric** and SOA migration plan in an iterative manner as the program gains migration experience and conditions change.

- BP1400: Programs will use authoritative **metadata** established by the Joint Mission Threads (JMTs) when available.

- BP1686: Align Node interfaces to **Components** for directory services with the guidance being provided by the Joint Enterprise Directory Services Working Group (JEDIWG) and sub-working groups, including such guidance as naming conventions, federation, and synchronization.

P1277: Design Tenet: Operations and Management of Transport and Services

This tenet encompasses three equally important principles of **Network Operations** (**NetOps**):

- Develop manageable systems
- Use non-proprietary implementations
- Use accepted industry standards

NetOps:

- Is a coordinated, comprehensive set of operational concepts and structure that fuses Systems and Network Management, Information Assurance/Computer Network Defense, and Content Staging/Information Dissemination Management into a single integrated operational construct
- Is an end-to-end capability that represents the integrated doctrine, force structure, and tactics, techniques, and procedures (TTP) needed to manage and direct the net-centric operations of the **Global Information Grid** (**GIG**)
- Encompasses all activities directly associated with the net-centric management and protection of GIG computing (including applications and systems), communications, and information assurance assets across the continuum of military operations
- Actively integrates those capabilities with the goal of end-to-end, assured network availability, information delivery, and information protect

Considerations

Develop Manageable Systems

- Build transport communications and network systems, services, sub-systems, sub-services, components, devices, and elements from the ground up to be "manageable." They should also have the appropriate functional management capabilities.
- Manage transport communications and network services and systems proactively and operate to specific levels of service. These service levels are documented and published in Operational or Service Level Agreements (OLA/SLAs).
- Fully integrate management solutions for transport systems and services with management solutions to ensure that the GIG is holistically operated and managed to support operational warfighter requirements. Operational management solutions should fully address all specific management functional areas; e.g., fault, configuration, accounting, performance, and security management.

Use Non-Proprietary Implementations

- Base operational management capabilities and solutions on non-proprietary implementations of industry accepted standards. An example is the Simple Network Management Protocol (SNMP) for IP-based networks.
- Critical transport systems, subsystems, component, and elements need to be able to monitor securely, detect changes in, and report the following:
    - Basic up/down operational status

- Performance information
- Operational configuration
- Security status

- Management interfaces should be non-proprietary. They must be accessible to a wide variety of management products and solutions via open-standards-based interfaces. The interfaces should not require hard-coding to obtain operational status information about a particular system.

- To support the development of NetOps Situational Awareness capabilities, ensure that operational management solutions can share operational status and other types of management information with management solutions operated by other types of service providers. The exchange must use non-proprietary standards-based interfaces. While this could be as simple as offering a browser-accessible Web interface using HTTP or HTTPS, management product vendors are beginning to implement Web services interfaces that use SOAP to share information between management systems.

Use Accepted Industry Standards and Emerging NetOps Concepts

- Operational concepts, architectures, processes, and procedures used by transport communications and network providers must incorporate emerging NetOps concepts. They should be based on accepted industry standards.

- Take an active role in the growing NetOps community. Develop the operational policies, processes, and procedures that enhance the flow of information between different management domains. This will ensure proactive problem detection, isolation, and resolution with minimum impact on the user. [R1262]

- To support this goal, adopt and implement operational policies, processes, and procedures based on internationally accepted de facto Telecommunication Service Provider and IT Service Management (ITSM) standards.

Support Standardized DoD Service-Oriented Environment

- Employ DoD-adopted standards for implementing and using transport infrastructure in the GIG-ES Enterprise Service Management (ESM)/NetOps service-oriented environment, rather than a domain or system-oriented environment.

- A Working Group established early in CY2003 to help develop DoD-level policy for operating in a service-oriented environment is co-chaired by ASD(NII)/DoD CIO and DISA. This group has enjoyed wide participation and representation from across the Services as well as from key enterprise programs. The main focus of this group has been to formulate initial ESM/NetOps requirements for GIG-ES and for the Net-Centric Enterprise Services (NCES) Program. The group also identified DoD-level policy areas that may need to be revised to support net-centric operations in a service-oriented architecture (SOA). In addition, the group has collaborated with the NetOps CONOPS group to broaden the current transport- and network-centric approach to one that is more holistic and consistent in monitoring, managing, and controlling systems, services, and applications, in addition to transport systems and networks.

Employ DoD-Adopted Standards to Support Cross-System and Domain Management

- Employ DoD-adopted standards for operating and managing transport services. This includes interaction with counterparts in other networks or management domains, such as system or application managers.

- Specify interfaces and/or standards for the following:
    - Sharing operational status and performance information
    - Collecting and disseminating service management information
    - Selecting the format in which it is made available (e.g., SNMP, XML, CIM, SOAP)

Note: *Volume 1 of the DISR [R1179] identifies SNMP and XML as mandated standards and CIM as an emerging standard; the NCOW RM [R1179] identifies CIM as a target standard.*

Plan for Coalition Interoperability

- Plan for operations and management of transport services. This includes interacting with counterparts in other networks or management domains used by coalition partners. Most recent conflicts have involved not only U.S. forces, but forces from allies and coalition partners. In the future, U.S. information and communications systems must support interoperability with these groups. There are various ways to achieve interoperability including the following:
    - Acquisition of common systems
    - Development of diverse but interoperable systems
    - Adherence to standards and commercial best practices

P1307: Open Technology Development

The Deputy Under Secretary of Defense (DUSD) for Advanced Systems and Concepts (AS&C) chartered the development of the OSD ***Open Technology Development Roadmap***. [R1288] The roadmap proposes that DoD adopt generally understood OTD practices regarding open source code access, open interfaces and systems, and collaborative development methodologies. The goal is to keep pace with technology advances and changing requirements in an efficient manner.

There are five aspects associated with OTD:

- Open Architecture
- Open Standards
- Open Development Collaboration
- Open Source (Software)
- Open Systems

P1309: Open Architecture

Open Architecture

Open Architecture (OA), according to ***Open Architecture Principles and Guidelines***, [R1288] is a pattern of nonfunctional requirements that contribute to the ability to create, deploy and manage OA systems. In some domains, e.g. systems engineering, OA considerations would apply to both hardware and software components. An Open Architecture employs open standards for key interfaces within a system [Open Systems Joint Task Force]. Open Architecture is the confluence of business and technical practices yielding modular, interoperable systems that adhere to open standards with published interfaces. This approach significantly increases opportunities for innovation and competition, enables reuse of components, facilitates rapid technology insertion, and reduces maintenance constraints. OA delivers increased warfighting capabilities in a shorter time at reduced cost [Naval Open Architecture Rhumb Lines; Open Architecture 12 Dec 06.pdf].

For an architecture to be "open" it must meet all of the following criteria.

> ***Note:*** *Specific terms are defined in Sections 2.1.2 through 2.1.7 of the* ***Open Architecture Principles and Guidelines****; links to applicable NESI Perspectives are in brackets following each question.*

- Modular
  - Is the architecture partitioned into discrete, self-contained modules of functionality?
    - [NESI on Implementing a Component-Based Architecture]
  - Do each of the modules have well defined, published interfaces?
    - [NESI on Public Interface Design]
    - [NESI on Standard Interface Documentation]
    - [NESI on how to Publish and Insulate Public Interfaces]
    - [NESI on Key Interface Profiles (KIPs)]
  - Are the interface definitions designed for ease of understanding by third-party architects?
    - [NESI on Exposing Functionality through Non-Standard Interfaces]
- Interoperable
  - Do the architecture modules enable the useful exchange of data and information with other systems outside of the architecture?
    - [NESI on Net-Centric Information Engineering]
  - Does each architecture module provide for the execution of its capabilities in response to requests coming from outside the respective module?
    - [NESI on the Software Communication Architecture (SCA)]
    - [NESI on Node Application Enterprise Services]
    - [NESI on Phases of SOA Adoption]

- Does each architecture module provide for the request for execution of capabilities that are instantiated outside of the respective module?
    - [NESI on Common Enterprise Services Definitions and Status]
- Extensible
    - Is the architecture designed with points of integration (e.g. module interfaces) that allow for future modules and capabilities to be added to the implementation, without requiring a modification to the architecture or existing implementation?
        - [NESI on Implementing Component-Based Architectures]
- Reuseable
    - Is the architecture designed with modules that can be used in multiple contexts to provide similar capabilities in those different contexts?
        - [NESI Pattern for Re-Implementation]
        - [NESI Contracting Guidance for Reuse]
- Composable
    - Is the architecture comprised of modules that can be selected and assembled in various combinations to satisfy specific user requirements?
        - [NESI on Implementing a Component-Based Architecture]
- Maintainable
    - Can the architecture's modules be maintained (revised, repaired, and replaced) without impacting the prescribed requirements (performance, availability, etc.) of the architecture's other modules?
        - [NESI on Management Issues for Exposed Functionality]
        - [NESI on Maintaining the Internal Component Environment]

P1310: Open Standards

The DoD Open Systems Joint Task Force defines Open Standards as standards that are widely used, consensus-based, published, and maintained by recognized standards organizations [OSJTF Terms & Definitions]. For a standard to be "open," it must meet the follow criteria:

- Is the standard widely-used?
- Is the standard consensus-based (developed using an open consortium approach)?
- Is the standard maintained and recognized by one or more recognized standards organizations, such as the Internet Society (ISOC), the Object Management Group (OMG), the Organization for the Advancement of Structured Information Standards (OASIS), or the World Wide Web Consortium (W3C)?
- Does each standard include all details necessary for interoperable implementation?
- Is the standard freely and publicly available under royalty-free terms?
- Are all patents to the implementation of the standard licensed under royalty-free terms for unrestricted use or covered by a promise of non-assertion when practiced by open source software?
- Is the standard free of all requirements for execution of a license agreement, non-disclosure agreement, grant, click-through arrangement, or any form of paperwork, to deploy conforming implementations of the standard?
- Is the standard free of all requirements for other technology that fails to meet this "open standard" criteria?

P1311: Open Development Collaboration

Open Development Collaboration is a team-based process to design, acquire, implement, deploy, and utilize a system. Include appropriately qualified subject matter experts from both government and industry, and include representatives of all stakeholders involved in the acquisition, deployment, and utilization of the system. Document the team's collaboration, correspondence, and decisions using an on-line mechanism (e.g. Web-based forum) that provides persistence and read/ write access to all team members; the government should retain all rights to the content placed in the on-line mechanism. The government may restrict access to this content to members of the respective team, as may be deemed necessary by the government representatives. The development collaboration is "open" if it meets all of the following criteria:

- Does the collaboration cover all aspects of the development lifecycle including design, acquisition, implementation, deployment, and utilization?
- Is the team that is collaborating comprised of appropriately qualified subject matter experts from both government and industry?
- Does the team that is collaborating include representatives of all stakeholders involved in the acquisition, deployment, and utilization of the system?
- Are the team's collaboration, correspondence, and decisions persistently documented using an on-line mechanism (such as forums)?
- Is that content/documentation freely accessible to all team members?
- Do all team members have read/write access to that documentation (and is the integrity of each team member's input perserved)?
- Does the government have full rights to that content?

Examples of Open Development Collaboration

- Source Forge - example of an open development collaboration site on the Internet
- NESI Collaboration Site - example of a development collaboration site with controlled access for authorized government users, contractors, and vendors
- TBMCS's DEVnet - example of a development collaboration site with controlled access for authorized government users, contractors, and vendors

P1312: Open Source (Software)

The principle of "Open Source" does not just mean access to the source code is freely and publicly available. The ***Open Source Initiative*** Open Source Definition includes ten criteria which form the basis of the following questions (note that links to applicable NESI Perspectives are in brackets after some of the questions). For software to meet the definition of "open source" it must satisfy the ten criteria.

- Is the license free of all restrictions (e.g., all royalties and other such fees for sale or use) preventing the DoD from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources?
  - [NESI Contracting Guidance for Acquisition]
  - [NESI Contracting Guidance for Reuse]
  - [NESI Guidance for Representations, Certifications, and other Statements of Offerors]
- Does the program include source code and allow for distribution of that source code in textual form as well as in compiled form?
  - [NESI Guidance for Standard Interface Documentation]
  - [NESI Guidance for RFP Section J - List of Attachments]
- Does the license allow for modifications and derived works, and allow those changes to be distributed under the same terms as the license of the original software?
- Does the license protect the integrity of the author's original source code? For example,
  - requiring derived works to carry a different name or version number from the original software?
  - requiring that the original source code be distributed as pristine based sources plus patches, so that "unofficial" changes (those made and added to the source by parties other than the original author) can be made available but easily distinguished from the base source?
- Is the license free from all restrictions which discriminate against any person or group of persons? (External policy might place such restrictions.)
- Is the license free from all restrictions that would prevent anyone from making use of the software in a specific field or endeavor?
- Are the rights attached to the software applicable to all whom the software is redistributed without the need for execution of an additional license by those parties?
- Are the rights attached to the software free from all dependencies on the software's being part of a particular software redistribution? (If the software is extracted from that distribution and used or distributed within the terms of the software's license, all parties to whom the software is redistributed should have the same rights as those granted in conjunction with the original software distribution.)
- Is the license free from all restrictions on other software that is distributed along with the licensed software? (For example, the license must not insist that all other software distributed on the same medium must be open source software.)
- Is the license free of all provisions that may be predicated on any individual technology or style of interface? (The license must be technology-neutral.)

P1313: Open Systems

The DoD Open Systems Joint Task Force (OSJTF) defines an open system as "a system that employs modular design, uses widely supported and consensus based standards for its key interfaces, and has been subjected to successful validation and verification tests to ensure the openness of its key interfaces" [OSJTF What is an Open System?]. The Acquisition Community Connection, hosted by the Defense Acquisition University, has additional information concerning Modular Open Systems Approach (MOSA), the DoD "open systems" implementation [ACC Community Browser].

The Carnegie Mellon University Software Engineering Institute further defines an open system as a collection of interacting software, hardware, and human components designed to satisfy stated needs with interface specifications of its components that are fully defined, available to the public and maintained according to group consensus in which the implementations of the components conform to the interface specifications [SEI Glossary].

For a system to be considered "open" it must meet all of the following criteria:

- Is the system based on an Open Architecture?
- Does the system employ Open Standards for its key interfaces?
- Are the system's key interfaces maintained using an Open Development Collaboration process?
- Are the system's key interfaces fully defined and available to the public, as is the case with Open Source?

P1279: Naval Open Architecture

Interoperability, Maintainability, Extensibility, Composeability, and Reuseability are non-functional requirements (NFRs) that support Open Architecture according to the ***Open Architecture Principles and GuidelinesR1184,R1307*** [R1307] which defines two types of relationships between NFRs, ***Enabled By*** and ***Facilitated By***. Enabled by is a strict dependence between NFRs while an NFR that facilitates another NFR is not required but contributes.

Below is the relationship between the NFRs

	Enabled By	***Facilitated By***
Interoperability		Open Standards
Maintainability		Composeability
		Reuseability
Extensibility	Modularity	Interoperability
Composeability	Reuseability	
Reuseability	Interoperability	
	Extensibility	

Detailed Perspectives

- Interoperability
- Maintainability
- Extensibility
- Composeability
- Reusability

P1280: Interoperability

Naval Open Architecture (OA) defines **interoperability** as being facilited by **Open Standards** which makes capabilities of a system a known quantity. OA does not restrict interoperability to the use of Open Standards.

Enablers of interoperability include the following:

- Well designed and document key internal interfaces
- Accessible metadata repository for syntatic interoperability
- **COI** established and standardized datamodels and metadata
- Availability of data
- Web service discovery
- Enterprise wide information assurance practicies
- Producer and consumer decoupling through message or event-drivn service bus

Inhibitors to interoperability include the following:

- Proprietary and/or unpublished APIs
- Point to point connectivity
- Application data models elevated to Enterprise data models
- Fine-grained service calls

Guidance

- G1001: Use formal standards to define public **interfaces**.
- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.
- G1005: Separate infrastructure capabilities from **mission** functions.
- G1007: Ensure that applications use open, standardized, **vendor**-neutral **API**(s).
- G1008: Isolate platform-specific **interfaces** and **vendor** dependencies.
- G1011: Make components independently deployable.
- G1012: Use a set of services to expose **Component** functionality.
- G1018: Assign version identifiers to all public interfaces.
- G1035: Follow W3C standards for code which will generate a Web page display.
- G1071: Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).
- G1073: Isolate vendor extensions to enterprise-services standard interfaces.

- G1078: Document the use of non-**Java EE**-defined **deployment descriptors**.
- G1080: Adhere to the **Web Services Interoperability Organization** (**WS-I**) Basic Profile specification for **Web service** environments.
- G1084: Validate documents transferred using **SOAP** against the **W3C XML** Standard by an **XML Schema Definition** (**XSD**) defined by the **Community of Interest** (**COI**).
- G1085: Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.
- G1093: Implement exception handlers for **SOAP**-based **Web services**.
- G1101: Use **Web services** to bridge **Java EE** and **.NET**.
- G1125: Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.
- G1127: Use a **UDDI** specification that supports publishing discovery services.
- G1131: Use industry standard Universal Description, Discovery, and Integration (**UDDI**) **APIs** for all UDDI inquiries.
- G1132: Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement the **SQL** standard.
- G1141: Use standard **data models** developed by **Communities of Interest** (**COI**) as the basis of program or project data models.
- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1209: For Java, use **JDK** logging facilities.
- G1210: For **.NET**, use Debug and Trace from the `System.Diagnostics` **namespace**.
- G1225: Use a build tool that is independent of the **Integrated Development Environment**.
- G1236: Do not **hard-code** the **endpoint** of a **Web service** vendor.
- G1237: Do not **hard-code** the configuration data of a **Web service** vendor.
- G1245: Isolate the **Web service** **portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.
- G1267: Use industry standard HTML data entry fields on Web pages.
- G1268: Label all data entry fields.
- G1270: Include scroll bars for text entry areas if the data buffer is greater than the viewable area.
- G1276: Do not modify the contents of the Web browser's status bar.
- G1277: Do not use tickers on a Web site.
- G1278: Use the browser default setting for links.
- G1284: Use only one font for **HTML** body text.
- G1285: Use **relative font sizes**.

- G1286: Provide text labels for all buttons.
- G1287: Provide feedback when a transaction will require the user to wait.
- G1292: Use text-based Web site navigation.
- G1293: Use descriptive labels for all clickable graphics.
- G1294: Provide a site map on all Web sites.
- G1295: Provide redundant text links for images within an **HTML** page.
- G1300: Secure all **endpoints**.
- G1301: Practice layered security.
- G1302: Validate all inputs.
- G1304: Unit test all code.
- G1305: Ensure the separation of **encrypted** and unencrypted information.
- G1306: **Identify** and **authenticate** users of the application.
- G1308: Configure **Public Key Enabled** applications to use a **Federal Information Processing Standard** (**FIPS**) 140-2 certified cryptographic module.
- G1309: Make applications handling high value unclassified information in Minimally Protected environments **Public Key Enabled** to interoperate with **DoD High Assurance** .
- G1310: Protect application cryptographic objects and functions from tampering.
- G1311: Use **Hypertext Transfer Protocol over Secure Socket Layer** (**HTTPS**) when applications communicate with DoD **Public Key Infrastructure** (**PKI**) components.
- G1312: Make applications capable of being configured for use with DoD **PKI**.
- G1314: Provide applications the ability to import and export keys (software certificates only).
- G1315: For applications, use key pairs and **Certificates** created for individuals using DoD **PKI** methods and procedures defined by the DoD Class 3 Public Key Infrastructure Interface Specification and the Personal Information Exchange Syntax Standard.
- G1316: Ensure that applications protect **private keys**.
- G1317: Ensure applications store **Certificates** for subscribers (the owner of the **Public Key** contained in the Certificate) when used in the context of signed and/or encrypted email.
- G1318: Develop applications such that they provide the capability to manage and store **trust points** (**Certificate Authority** Public Key **Certificates**).
- G1319: Ensure applications can recover data encrypted with legacy keys provided by the DoD **PKI** Key Recovery Manager (**KRM**).
- G1320: Use a minimum of 128 bits for **symmetric keys**.
- G1321: Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.

- G1322: Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of encrypting and decrypting data using the **Triple Data Encryption Algorithm** (**TDEA**).

- G1323: Generate random **symmetric encryption** keys when using symmetric encryption.

- G1324: Protect **symmetric keys** for the life of their use.

- G1325: Encrypt **symmetric keys** when not in use.

- G1326: Ensure applications are capable of producing Secure Hash Algorithm (**SHA**) **digests** of **messages** to support verification of DoD **PKI** signed objects.

- G1327: Enable an application to obtain new **Certificates** for subscribers.

- G1328: Enable an application to retrieve **Certificates** for use, including relying party operations.

- G1330: Ensure applications are capable of checking the status of **Certificates** using a **Certificate Revocation List** (**CRL**) if not able to use the **Online Certificate Status Protocol** (**OCSP**).

- G1331: Ensure applications are able to check the status of a Certificate using the **Online Certificate Status Protocol** (**OCSP**).

- G1333: Only use a **Certificate** during the Certificate's validity range, as bounded by the Certificate's "Validity - Not Before" and "Validity - Not After" date fields.

- G1335: Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.

- G1338: Applications and **Certificates** need to be able to support multiple organizational units.

- G1339: Practice defensive programming by checking all method arguments.

- G1341: Use a security manager support to restrict application access to privileged system resources.

- G1343: Declare classes final to stop inheritance and prevent methods from being overridden.

- G1344: Encrypt sensitive data stored in configuration or resource files.

- G1347: Secure remote connections to a database.

- G1349: Validate all input that will be part of any dynamically generated **SQL**.

- G1350: Implement a strong password policy for **RDBMS**.

- G1351: Enhance database security by using multiple user accounts with constraints.

- G1352: Use database clustering and redundant array of independent disks (RAID) for high availability of data.

- G1356: Use the **SOAP** standard for all **Web services**.

- G1357: Do not rely solely on transport level security like **SSL** or **TLS**.

- G1359: Bind **SOAP Web service** security policy assertions to the service by expressing them in the associated **WSDL** file.

- G1362: Validate incoming XML-based messages using a **schema**.

- G1363: Do not use clear text passwords.

- G1364: Hash all passwords using the combination of a timestamp, a **nonce** and the password for each **message** transmission.
- G1365: Specify an expiration value for all security tokens.
- G1366: Digitally sign all **messages** where non-repudiation is required.
- G1367: Digitally sign **message** fragments that are required not to change during transport.
- G1369: Digitally sign all requests made to a security token service.
- G1371: Use the **Digital Signature Standard** for creating **Digital Signatures**.
- G1372: Use an X.509 **Certificate** to pass a **Public Key**.
- G1373: **Encrypt messages** that cross an **IA** boundary.
- G1374: Individually **encrypt** sensitive **message** fragments intended for different intermediaries.
- G1376: Do not **encrypt** key elements that are needed for correct **SOAP** processing.
- G1377: Use **LDAP** 3.0 or later to perform all connections to LDAP repositories.
- G1378: Encrypt communication with **LDAP** repositories.
- G1379: Use **SAML** version 2.0 for representing security assertions.
- G1380: Use the **XACML** 2.0 standard for **SAML**-based rule engines.
- G1381: Encrypt all sensitive persistent data.
- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1569: Maintain a comprehensive list of all of the **Components** that are part of the Node.
- G1570: Assume an active management role among the **Components** within the Node.
- G1581: Expose **legacy system** or **application** functionality through the use of a service that uses a **facade design pattern**.
- G1635: Make Nodes that will be part of the **Global Information Grid** (GIG) consistent with the *GIG Integrated Architecture*.

- G1636: Comply with the **Net-Centric Operations and Warfare Reference Model** (NCOW RM).
- G1637: Make Node-implemented **directory services** comply with the directory services **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs).
- G1638: Comply with the directory services **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node directory services **proxies**.
- G1640: Register **Components** exposed by the Node with the **DISA**-hosted registries.
- G1641: Comply with the Service Discovery **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node-implemented **Service Discovery** (SD).
- G1642: Comply with the **Service Discovery Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node Service Discovery (SD) **proxies**.
- G1644: Comply with the **Federated Search** # **Search Web Service** (SWS) **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node implemented Federated Search # Search Web Service (SWS).
- G1645: Implement a local **Content Discovery Service** (CDS).
- G1646: Comply with the directory services **Global Information Grid** (**GIG**) **Key Interface Profiles (KIPs**) in Node **Federated Search** Services **proxies**.
- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.
- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.
- G1724: Develop XML documents to be well formed.
- G1725: Develop XML documents to be **valid** XML.
- G1726: Define XML Schemas using **XML Schema Definition** (XSD).
- G1727: Provide names for XML type definitions.
- G1728: Define types for all **XML elements**.
- G1730: Follow an XML coding standard for defining schemas.
- G1737: Define a target namespace in schemas.
- G1746: Develop XSLT stylesheets that are XSLT version agnostic.
- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1754: Give each new XML schema version a unique URL.
- G1759: Use a style guide when developing Web portlets.
- G1761: Provide units of measurements when displaying data.
- G1763: Indicate the security classification for all classified data.
- G1770: Explicitly define the **Data Distribution Service** (DDS) **Domains** for the system.

- G1771: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of a **publisher**.
- G1772: Assign a unique identifier for each **Data-Distribution Service** (DDS) **Domain** within the system.
- G1785: Stipulate that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on reuse of common functionality.
- G1786: Stipulate that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on well defined services.
- G1787: Stipulate that the Offeror is to use NESI to assess net-centricity and interoperability.
- G1796: Explicitly define all the **Data Distribution Service** (DDS) **Domain Topics**.
- G1798: Explicitly define all the **Data Distribution Service** (DDS) **Domain data types**.
- G1799: Explicitly associate data types to the **Data Distribution Service** (DDS) **Topics** within a DDS **Domain**
- G1800: Explicitly identify Keys within the **Data Distribution Service** (DDS) **data type** that uniquely identify an instance of a data object.
- G1801: Explicitly define a **Topic Quality of Service** (QoS) for each **Data Distribution Service** (DDS) Topic within a DDS **Domain**.
- G1803: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe real-time messaging criteria for **Publishers**.
- G1804: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe **DataWriter**.
- G1805: Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of the **Subscriber**.
- G1806: Explicitly define the Request-Offered **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of the **DataReader**.
- G1808: Handle all **Data Distribution Service** (DDS) **Quality of Service** (QoS) contract violations using one of the **Subscriber access APIs**.
- G1810: Use **data models** to document the data contained within the **Data Distribution Service** (DDS) **Data-Centric Publish Subscribe** (DCPS).
- G1797: Use a minimum of 1024 bits for **asymmetric keys**.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.

P1281: Maintainability

In the Naval Open Architecture (OA) context, maintainability is "the portion of a component's or sytem's lifecycle after installation, including its end of life. Key to this lifecycle is updating the system to introduce new technology, changed business processes, etc." (see ***Open Architecture Principles and Guidelines*** section 2.1.7.1 [R1307]). Maintainability depends on a modular system with well-defined interfaces and documentation for all aspects of the lifecycle of a system.

Enablers of maintainability include the following:

- Modular design with well-defined, stable interfaces
- Loose coupling
- Clear and concise documentation
- Use cases and testing
- Compliance with open standards

Inhibitors of maintainability include the following:

- Frequent changes to interfaces
- Tightly coupled and heavily optimized solutions

Guidance

- G1001: Use formal standards to define public **interfaces**.
- G1002: Separate public **interfaces** from implementation.
- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.
- G1004: Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.
- G1018: Assign version identifiers to all public interfaces.
- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1021: Create fully insulated classes.
- G1022: Insulate public **interfaces** from compile-time dependencies.
- G1027: Internally document all source code developed with DoD funding.
- G1032: Validate all input fields.
- G1043: Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.
- G1044: Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.
- G1052: Use the code-behind feature in ASP.NET to separate presentation code from the business logic.
- G1053: Do not embed HTML code in any code-behind code used by aspx pages.

- G1056: Specify a versioning policy for **.NET** assemblies.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1060: Encapsulate Java code that is used in **JSP**(s) in tag libraries.
- G1071: Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).
- G1073: Isolate vendor extensions to enterprise-services standard interfaces.
- G1082: Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).
- G1083: Do not pass **Web Services-Interoperability Organization** (**WS-I**) **Document Object Model** (**DOM**) documents as strings.
- G1085: Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.
- G1088: Use isolation design patterns to define system functionality that manipulates **Web services**.
- G1090: Do not **hard-code** a **Web service's endpoint**.
- G1094: Catch all exceptions for application code exposed as a **Web service**.
- G1095: Use **W3C** fault codes for all **SOAP** faults.
- G1118: Localize **CORBA** vendor-specific source code into separate **modules**.
- G1121: Do not modify **CORBA** Interface Definition Language (**IDL**) compiler auto-generated stubs and skeletons.
- G1132: Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement the **SQL** standard.
- G1146: Include information in the **data model** necessary to generate a **data dictionary**.
- G1147: Use **domain analysis** to define the constraints on input data validation.
- G1148: **Normalize** data models.
- G1151: Define declarative **foreign keys** for all relationships between tables to enforce **referential integrity**.
- G1153: Separate application, presentation, and data tiers.
- G1154: Use **stored procedures** for operations that are focused on the insertion and maintenance of data.
- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1204: Create configuration services to provide distributed user control of the appropriate configuration parameters.
- G1205: Use non-source code persistence to store all user-modifiable **CORBA** service configuration parameters.
- G1208: Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.
- G1213: Provide an architecture design document.
- G1214: Provide a document with a plan for **deprecating** obsolete **interfaces**.

- G1215: Provide a coding standards document.
- G1216: Provide a software release plan document.
- G1217: Develop and use externally configurable components.
- G1218: Use a build tool that supports operation in an automated mode.
- G1219: Use a build tool that checks out files from configuration control.
- G1220: Use a build tool that **compiles** source code and dependencies that have been modified.
- G1221: Use a build tool that creates libraries or archives after all required compilations are completed.
- G1222: Use a build tool that creates executables.
- G1223: Use a build tool that is capable of running unit tests.
- G1224: Use a build tool that cleans out intermediate files that can be regenerated.
- G1225: Use a build tool that is independent of the **Integrated Development Environment**.
- G1236: Do not **hard-code** the **endpoint** of a **Web service** vendor.
- G1237: Do not **hard-code** the configuration data of a **Web service** vendor.
- G1239: Use design patterns (e.g., **facade**, **proxy**, or **adapter**) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.
- G1267: Use industry standard HTML data entry fields on Web pages.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1283: Use **linked style sheets** rather than embedded styles.
- G1300: Secure all **endpoints**.
- G1301: Practice layered security.
- G1307: Provide a security policy file.
- G1308: Configure **Public Key Enabled** applications to use a **Federal Information Processing Standard** (**FIPS**) 140-2 certified cryptographic module.
- G1309: Make applications handling high value unclassified information in Minimally Protected environments **Public Key Enabled** to interoperate with **DoD High Assurance** .
- G1311: Use **Hypertext Transfer Protocol over Secure Socket Layer** (**HTTPS**) when applications communicate with DoD **Public Key Infrastructure** (**PKI**) components.
- G1312: Make applications capable of being configured for use with DoD **PKI**.
- G1313: Provide documentation for application configuration and setup for use with DoD **PKI**.
- G1314: Provide applications the ability to import and export keys (software certificates only).
- G1315: For applications, use key pairs and **Certificates** created for individuals using DoD **PKI** methods and procedures defined by the DoD Class 3 Public Key Infrastructure Interface Specification and the Personal Information Exchange Syntax Standard.

- G1318: Develop applications such that they provide the capability to manage and store **trust points** (**Certificate Authority** Public Key **Certificates**).

- G1319: Ensure applications can recover data encrypted with legacy keys provided by the DoD **PKI** Key Recovery Manager (**KRM**).

- G1320: Use a minimum of 128 bits for **symmetric keys**.

- G1321: Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.

- G1322: Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of encrypting and decrypting data using the **Triple Data Encryption Algorithm** (**TDEA**).

- G1323: Generate random **symmetric encryption** keys when using symmetric encryption.

- G1324: Protect **symmetric keys** for the life of their use.

- G1325: Encrypt **symmetric keys** when not in use.

- G1326: Ensure applications are capable of producing Secure Hash Algorithm (**SHA**) **digests** of **messages** to support verification of DoD **PKI** signed objects.

- G1327: Enable an application to obtain new **Certificates** for subscribers.

- G1328: Enable an application to retrieve **Certificates** for use, including relying party operations.

- G1330: Ensure applications are capable of checking the status of **Certificates** using a **Certificate Revocation List** (**CRL**) if not able to use the **Online Certificate Status Protocol** (**OCSP**).

- G1331: Ensure applications are able to check the status of a Certificate using the **Online Certificate Status Protocol** (**OCSP**).

- G1333: Only use a **Certificate** during the Certificate's validity range, as bounded by the Certificate's "Validity - Not Before" and "Validity - Not After" date fields.

- G1335: Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.

- G1338: Applications and **Certificates** need to be able to support multiple organizational units.

- G1340: Log all exceptional conditions.

- G1342: Restrict direct access to class internal variables to functions or methods of the class itself.

- G1343: Declare classes final to stop inheritance and prevent methods from being overridden.

- G1346: Audit database access.

- G1348: Log database **transactions**.

- G1352: Use database clustering and redundant array of independent disks (RAID) for high availability of data.

- G1356: Use the **SOAP** standard for all **Web services**.

- G1359: Bind **SOAP Web service** security policy assertions to the service by expressing them in the associated **WSDL** file.

- G1372: Use an X.509 **Certificate** to pass a **Public Key**.

- G1378: Encrypt communication with **LDAP** repositories.
- G1576: Provide an environment to support the development, build, integration, and test of net-centric capabilities.
- G1577: Maintain an **Enterprise Service** schedule for interim and final **enterprise** capabilities within the Node.
- G1578: Define a schedule for **Components** that includes the use of the **Enterprise Services** defined within the Node's enterprise service schedule.
- G1582: In Node **Enterprise Service** schedules, include version numbers of standard Enterprise Services interfaces being implemented.
- G1583: Provide routine **Enterprise Services** schedule updates to every **Component** of a Node.
- G1717: Use constants instead of hard-coded numbers for characteristics that may change throughout the lifetime of the model.
- G1718: Design circuits to be synchronous.
- G1719: Automate testbench error checking in VHDL development.
- G1727: Provide names for XML type definitions.
- G1728: Define types for all **XML elements**.
- G1729: Annotate XML type definitions.
- G1730: Follow an XML coding standard for defining schemas.
- G1731: Only reference **XML elements** defined by a Type in substitution groups.
- G1735: Use the **`.xsd`** file extension for files that contain XML Schema definitions.
- G1736: Separate document schema definition and document instance into separate documents.
- G1740: Append the suffix Type to XML type names.
- G1744: Only reference abstract **XML elements** in substitution groups.
- G1745: Append the suffix Group to substitution group **XML element** names.
- G1751: Document all XSLT code.
- G1753: Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.
- G1754: Give each new XML schema version a unique URL.
- G1755: Use accepted file extensions for all files that contain XSL code.
- G1756: Isolate XPath expression statements into the configuration data.
- G1773: Use **`#include`** guards for all headers.
- G1774: Make header files self-sufficient.
- G1775: Do not overload the logical **`AND`** operator.
- G1776: Do not overload the logical **`OR`** operator.

- G1777: Do not overload the **`comma`** operator.
- G1778: Place all **`#include`** statements before all namespace **`using`** statements.
- G1779: Explicitly namespace-qualify all names in header files.

P1282: Extensibility

Extensible systems facilitate adding future capabilities and points of contact or integration. To support this, Open Architecture defines an extensible system as one with "sufficient internal quality and compartmentalization of data and behavior that new capabilities do not introduce unintended chages to existing data and behavior" (see ***Open Architecture Principles and Guidelines*** [R1307]). To achieve this, a system must be modular and be interoperable.

Enablers of extensibility include the following:

- Well defined points of variability
- Layered architecture
- Loose coupling

Inhibitors to extensibility include the following:

- Undocumented design and architecture assumptions

Guidance

- G1002: Separate public **interfaces** from implementation.
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1271: Provide instructions and **HTML** examples for all style sheets.

P1283: Composeability

Composeable systems allow for components to be selected and assembled in different ways to meet user requirements. In order for a system to be composeable its components must also be reuseable, interoperable, extensible, and modular as defined by Open Architecture. [R1307]

Enablers of composeability include the following:

- Standard enterprise ontology
- Enterprise service bus
- Clearly defined QoS
- Tools for composing services

Inhibitors to composeability include the following:

- No enterprise architecture management

Guidance

- G1002: Separate public **interfaces** from implementation.
- G1003: Separate the contents of application libraries that are to be shared from libraries that are to be used internally.
- G1011: Make components independently deployable.
- G1012: Use a set of services to expose **Component** functionality.
- G1022: Insulate public **interfaces** from compile-time dependencies.
- G1045: Define **XML** format information separately in **XSL**.
- G1050: In **ASP**, isolate the presentation tier from the middle tier using **COM** objects.
- G1052: Use the code-behind feature in ASP.NET to separate presentation code from the business logic.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1060: Encapsulate Java code that is used in **JSP**(s) in tag libraries.
- G1088: Use isolation design patterns to define system functionality that manipulates **Web services**.
- G1144: Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.
- G1153: Separate application, presentation, and data tiers.
- G1155: Use **triggers** to enforce **referential** or **data integrity**, not to perform complex **business logic**.
- G1202: Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).
- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.

- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.

- G1719: Automate testbench error checking in VHDL development.

P1284: Reusability

Open Architecture defines a reusable artifact as one that provides a capability that can be used in multiple contexts. Reuse is not confined to a software component but any lifecycle artifact including training, documentation, and configuration. Open Architecture is concerned with artifacts which relate to the design, construction, and configuration of a component.

Enablers of reusability include the following:

- Use of Reuseable Asset Specification (RAS)
- Low code complexity
- Components that depend primarily on OA interfaces

Inhibitors to reusability include the following:

- Serialized or single-threaded implementation
- Proprietary standards
- Cut-and-paste programming

Guidance

- G1019: Deprecate public interfaces in accordance with a published deprecation policy.
- G1045: Define **XML** format information separately in **XSL**.
- G1058: Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.
- G1060: Encapsulate Java code that is used in **JSP**(s) in tag libraries.
- G1144: Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.
- G1203: Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.
- G1217: Develop and use externally configurable components.
- G1271: Provide instructions and **HTML** examples for all style sheets.
- G1283: Use **linked style sheets** rather than embedded styles.
- G1311: Use **Hypertext Transfer Protocol over Secure Socket Layer** (**HTTPS**) when applications communicate with DoD **Public Key Infrastructure** (**PKI**) components.
- G1321: Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.
- G1335: Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.
- G1356: Use the **SOAP** standard for all **Web services**.
- G1377: Use **LDAP** 3.0 or later to perform all connections to LDAP repositories.

- G1382: Be associated with one or more **Communities of Interest** (**COIs**).
- G1383: Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.
- G1384: Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.
- G1385: Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.
- G1386: Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.
- G1387: Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.
- G1388: Use predefined commonly used database tables in the **DoD Metadata Registry**.
- G1389: Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.
- G1569: Maintain a comprehensive list of all of the **Components** that are part of the Node.
- G1713: Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.
- G1714: Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.
- G1717: Use constants instead of hard-coded numbers for characteristics that may change throughout the lifetime of the model.
- G1718: Design circuits to be synchronous.
- G1719: Automate testbench error checking in VHDL development.
- G1759: Use a style guide when developing Web portlets.
- G1773: Use **`#include`** guards for all headers.
- G1774: Make header files self-sufficient.
- G1775: Do not overload the logical **`AND`** operator.
- G1776: Do not overload the logical **`OR`** operator.
- G1777: Do not overload the **`comma`** operator.
- G1778: Place all **`#include`** statements before all namespace **`using`** statements.
- G1779: Explicitly namespace-qualify all names in header files.
- G1784: Include a statement in the solicitation for Contractors to identify and list data rights for all proposed products.
- G1785: Stipulate that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on reuse of common functionality.
- G1786: Stipulate that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on well defined services.
- G1787: Stipulate that the Offeror is to use NESI to assess net-centricity and interoperability.

- G1788: Stipulate that the Offeror is to use Government approved data rights labels and markings for all deliverables that are identified as Unlimited or Government Purpose Rights.

Best Practices

- BP1392: Register services in accordance with a documented service registration plan.

P1122: Relationship with the JCIDS Process

The appropriate timeframe to start implementing net-centricity and interoperability is during the early definition of the system with the preparation of the Capabilities Documents. These documents, prepared under the **Joint Capabilities Integration and Development System** (**JCIDS**), set the stage for the subsequent acquisition process. Before initiating a program, the JCIDS process identifies warfighting capability and supportability gaps and the **Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, and Facilities** (**DOTMLPF**) capabilities required to fill those gaps. The documentation developed during the JCIDS process provides the formal communication of capability needs between the warfighter, acquisition, and resource management communities.

Program sponsors, in coordination with program managers, should consider applicable NESI guidance when preparing JCIDS documents. Program sponsors and managers can use Part 1 and Part 2 to develop a high-level foundational understanding of the relevant issues and have a starting point for planning relevant activities and strategies. Incorporating this guidance facilitates meeting the requirements of the ASD(NII) Net-Centric Checklist (see P1239). This is a means of increasing interoperability and aiding the development of architectural products. Program personnel should look for the attributes in the program capabilities documents (with reference to the relevant portions of NESI) that are contained in Table 1 below.

Table 1 - Relationship between JCIDS Documents, Process Milestones, and NESI Guidance

JCIDS Document	Milestones	Description	Relevant NESI Guidance
Initial Capabilities Document (ICD)	A, B, C	Defines capability gap in terms of functional area(s), relevant range of military operations, time, obstacles to overcome, and key attributes, with appropriate measures of effectiveness. Recommends materiel approach(s) based on cost analysis, efficacy, sustainability, environmental quality impacts, and associated risks.	Parts 1, 2
Capability Development Document (CDD)	B	Provides operational performance attributes, including supportability, for the acquisition community to design the proposed system. Includes key performance parameters (KPP) and other parameters that guide the development, demonstration, and testing of the current increment. Outlines the overall strategy for developing full capability.	Parts 2, 3, 4 Net-Ready Key Performance Parameter (NR-KPP) developed for this CDD
Capability Production Document (CPD)	C	Addresses the production attributes and quantities specific to a single	Parts 3, 4, 5

		increment of an acquisition program. Supersedes threshold and objective performance values of the CDD.	Updated NR-KPP required in this CPD

The Net-Ready Key Performance Parameter (NR-KPP) noted in Table 1 measures the net-centricity of a new program or major upgrade. The NR-KPP contains four elements:

- Compliance with the **Net-Centric Operations and Warfare Reference Model** (**NCOW RM**)
- Compliance with applicable Global Information Grid **Key Interface Profiles** (**KIPs**)
- Compliance with DoD **information assurance** (**IA**) requirements
- Support for integrated architecture products that assess information exchange and use for a given capability

Refer to the **Defense Acquisition University** (**DAU**) Defense Acquisition Guidebook Section 7.3.4 for further information on the NR-KPP elements.

The program sponsor and manager can also use NESI to aid in the development of the NR-KPP as show in Table 2.

Table 2 - Relationship between NESI and the NR-KPP

NESI	NCOW RM Services Strategy	NCOW RM Data Strategy	NCOW RM IA Strategy	Information Assurance	Key Interface Profiles (KIPs)	Integrated Architectures
Part 1	3.2, 3.3.2, 4.4	3.2, 3.4, 4.2	3.2		3.3.1	1.5, 4.3 - 4.6
Part 2	4.1, 4.7, 7.0, 8.0	3.1 - 3.6, 8.0	5.1 - 5.7, 8.0	5.1 - 5.7, 8.0	4.1	4.1, 4.2, 6.3
Part 3	All	Net-Centric Data Strategy (NCDS)	Migration Concern: Security			Migration Concern: Architecture Documentation Maintenance, Migration Planning Process
Part 4	2.2 - 2.4	2.2 - 2.4	2.2 - 2.4	2.2 - 2.4	2.2 - 2.4	All of Part 4, but especially 2.4 .1
Part 5	Web Services, Browser-Based Clients	Data Tier, Data, Metadata	Application Security	Application Security		Technical Guidance and Tactics
Part 6	N/A	N/A	N/A	N/A	N/A	N/A

Guidance and Best Practice Details

G1001

Statement:

Use formal standards to define public **interfaces**.

Rationale:

It is important to use a common language to define the interfaces so producers and consumers can work independently and together.

There are many standards for defining interfaces (**UML**, **WSDL**, and **CORBA**). Use a documented standard that is widely accepted by industry.

Referenced By:

Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Make Data Interoperable
Interoperability
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Publish and Insulate Public Interfaces

Evaluation Criteria:

1) Test: [G1001.1]

Do **UML** documents exist that describe the shared interfaces?

Procedure:

Ask for the design documents to be provided during the review process.

Example:

None

2) Test: [G1001.2]

Are there **WSDL** files that document the interface to Web services?

Procedure:

Look for the existence of `.WSDL` files.

Example:

None

3) Test: [G1001.3]

Are there **IDL** files that document the interfaces to **CORBA** services?

Procedure:

Look for the existence of **`.idl`** files.

Example:

None

G1002

Statement:

Separate public **interfaces** from implementation.

Rationale:

This guidance encourages clean separation between **interface** and implementation details for all types of application development. This allows components and systems to be **loosely coupled**. The flexibility allows groups of developers to work independently and in parallel to the contract defined by the interface.

Another benefit of hiding implementation details is that it allows the implementation to change without affecting users of the interface. This means the interface can support dynamic and pluggable implementation.

Referenced By:

Design Tenet: Open Architecture
Composeability
Publish and Insulate Public Interfaces
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Maintainability
Extensibility

Evaluation Criteria:

1) Test: [G1002.1]

C++: Check to make sure interfaces are defined as pure virtual functions.

Procedure:

Make sure **C++** classes are defined in header files. Classes that represent external interfaces should contain only pure virtual functions. Make sure the class does not declare non-constant data members. Also, make sure it does not define default implementation. An interface should provide no default behavior.

Example:

None

2) Test: [G1002.2]

C: Check to make sure functions are declared in a header file using prototypes.

Procedure:

Make sure each library function has a prototype declaration in the header file.

Example:

None

G1003

Statement:

Separate the contents of application libraries that are to be shared from libraries that are to be used internally.

Rationale:

The public libraries that are intended to be shared with outside consumers need to remain fairly static in order to facilitate independent development by the **consumer** and the **producer** of the libraries' functionality. The consumer and the producer should mutually agree to changes in libraries.

All library content should not have external dependencies that are not related to supporting the interface.

There must be clear separation between domain-specific and shared libraries. Libraries that will be used in joint or multiple projects should not have domain-specific code.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Interoperability
Maintainability
Publish and Insulate Public Interfaces
Composeability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Cross-Security-Domains Exchange

Evaluation Criteria:

1) Test: [G1003.1]

Do the publicly shared libraries have any private or undocumented functionality?

Procedure:

Check each library against the publicly defined header and make sure that all objects or methods are public.

Example:

None

2) Test: [G1003.2]

Does the library contain extraneous interfaces or code that is not required?

Procedure:

Use coverage tool/Junit to make sure there is no extraneous code.

Example:

None

3) Test: [G1003.3]

Do the publicly shared libraries have any private or undocumented functionality?

Procedure:

Check to make sure that one library use of another library does not cross domain-specific boundaries. For instance, a common library of utilities should not have dependencies on another library that supports a specific such as UHF satellites. However, the reverse is okay.

Example:

None

G1004

Statement:

Make public **interfaces** backward-compatible within the constraints of a published **deprecation** policy.

Rationale:

The public interface is basically a contract between the **producer** of the functionality defined in an interface and the **consumer** of the functionality. This and related guidance statements are intended to ensure that this contract remains intact and that the consumer of the functionality is not broken during the update cycle of the interface.

Referenced By:

Public Interface Design
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Maintainability
Design Tenet: Open Architecture
Publish and Insulate Public Interfaces
Versioning XML Schemas

Evaluation Criteria:

1) Test: [G1004.1]

Does the public interface (interfaces that are used externally, outside the project's domain) contain versioning information?

Procedure:

Check to make sure the interface/class has versioning information.

Example:

None

2) Test: [G1004.2]

Does the document structure contain a document that indicates the shelf life of deprecated interfaces?

Procedure:

Check for project documents that have information on the life of deprecated interfaces.

Example:

None

G1005

Statement:

Separate infrastructure capabilities from **mission** functions.

Rationale:

Applications should not try to reinvent the wheel by creating custom **enterprise services** such as messaging, directory services, logging, etc. Application development should use standardized **APIs** to access common enterprise services. For instance, in Java, use **JMS** to access a messaging system.

Referenced By:

Publish and Insulate Public Interfaces
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Interoperability

Evaluation Criteria:

1) Test: [G1005.1]

Does the application re-create common and available enterprise services?

Procedure:

Check the application code for code that recreates functionality of an enterprise service.

Example:

None

2) Test: [G1005.2]

Does the application code access enterprise services in a vendor-specific way?

Procedure:

Check for code that accesses a vendor-specific API instead of utilizing an industry-standard API.

Example:

None

G1007

Statement:

Ensure that applications use open, standardized, **vendor**-neutral **API**(s).

Rationale:

Using standardized, open APIs will enable the code to be more portable. It will also prevent vendor lock-in. "Standardized" means industry consensus. "Open" means available to everyone.

Referenced By:

Publish and Insulate Public Interfaces
Design Tenet: Open Architecture
Interoperability
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1007.1]

Does the application create customized/proprietary solutions where standardized **API**s exists?

Procedure:

Check the application for code that has proprietary solutions where standardized APIs exists. For instance, does the application write its own messaging system, bypassing utilizing the API.

Example:

None

2) Test: [G1007.2]

Does the application utilize vendor-specific **API**s?

Procedure:

Check the application to make sure it is not using vendor-specific APIs. For instance, see if the application accesses the database using a proprietary interface from Oracle instead of the standard calls.

Example:

None

G1008

Statement:

Isolate platform-specific **interfaces** and **vendor** dependencies.

Rationale:

Insulating platform-specific code using standard abstractions or custom classes will keep all non-portable code in one place and prevent proliferation of non-portable code throughout the application.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Publish and Insulate Public Interfaces
Design Tenet: Accommodate Heterogeneity
Interoperability

Evaluation Criteria:

1) Test: [G1008.1]

Does the application contain any platform-specific code that has not been abstracted?

Procedure:

Check code that is non-portable; for instance, the code does not use back slashes (Windows) or forward slashes (UNIX) in literal strings to create a path.

Example:

```
String path = "\tmp";
```

2) Test: [G1008.2]

Is platform-specific code isolated into a single class or file?

Procedure:

Search the files for platform-specific code.

Example:

None

G1010

Statement:

Use **open-standard** logging frameworks.

Rationale:

Standardizing on one logging **API** means the code will be more portable between developers, and developers no longer need to learn multiple logging frameworks.

Referenced By:

Publish and Insulate Public Interfaces
Design Tenet: Open Architecture
Design Tenet: Enterprise Service Management
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1010.1]

See sublevel guidance: G1209, G1210.

Procedure:

Example:

G1011

Statement:

Make components independently deployable.

Rationale:

Independently deployable components do not have any dependencies on other components. This is often unattainable because components are often aggregations of lower-level components. Exceptions to this rule can occur if the relationships between components are one or more of the following:

- well-defined and well thought out
- carefully managed
- externally configurable

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Design Tenet: Accommodate Heterogeneity
Composeability
Implement a Component-Based Architecture
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1011.1]

Is the component dependent on other components?

Procedure:

Check for dependencies.

Example:

None

G1012

Statement:

Use a set of services to expose **Component** functionality.

Rationale:

By exposing discrete units of functionality as **services**, business and data integrity remain intact. A service receives a request, processes it, and returns the result to the requester as a single operation.

Referenced By:

Design Tenet: Scalability
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Implement a Component-Based Architecture
Design Tenet: Accommodate Heterogeneity
Design Tenet: Open Architecture
Composeability

Evaluation Criteria:

1) Test: [G1012.1]

Are there **WAR** files that contain the component?

Procedure:

Check for the occurrence of **`.war`** files.

Example:

None.

2) Test: [G1012.2]

Are there **WSDL** files that define the services?

Procedure:

Check for the occurrence of **`.wsdl`** files.

Example:

None.

G1014

Statement:

Access databases through **open standard** interfaces.

Rationale:

The use of non-standard interfaces can cause portability issues. Standards-based database interfaces promote database independence. For example, **ODBC** is a standard database interface for referencing databases with C/C++ and .NET, while Java Database Connection (**JDBC**) is a standard **API** for accessing databases with Java.

Referenced By:

Decouple from Applications
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1014.1]

Are standard interfaces used to access databases?

Procedure:

Check that standards-based interfaces are used to access databases; for example, ODBC for C,C++, or .NET languages, or JDBC for Java.

Example:

None.

G1018

Statement:

Assign version identifiers to all public interfaces.

Rationale:

Assigning versions is necessary when determining compatibility between the **interface** and its **consumer**. Versioning public interfaces allows all parties to track the evolution of the interface for backward compatibility. This can help consumers plan for integration and migration. It is important to have the version information in the shared public interface code because it identifies the actual interface to which consumers of the interface will be coding. Another benefit is that it allows tools to generate the documentation automatically so it does not need to be in two places.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
Public Interface Design
Interoperability
Publish and Insulate Public Interfaces
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1018.1]

Does the shared public interface code contain versioning information?

Procedure:

Inspect public interfaces or their supporting documentation for version identifiers.

Example:

None.

G1019

Statement:

Deprecate public interfaces in accordance with a published deprecation policy.

Rationale:

By deprecating instead of removing interfaces, development teams can plan for software migration and continue to run the software with existing (but deprecated) interfaces.

Referenced By:

Reusability
Public Interface Design
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Publish and Insulate Public Interfaces
Design Tenet: Accommodate Heterogeneity
Versioning XML Schemas
Maintainability

Evaluation Criteria:

1) Test: [G1019.1]

Are public interfaces appropriately deprecated?

Procedure:

Check the project documentation for deprecation policy.

Check that interfaces are properly marked and removed according to the deprecation policy.

Example:

None

G1021

Statement:

Create fully insulated classes.

Rationale:

Data members should not be public.

Do not expose implementation details of a class. For instance, information such as the use of a link list or hashtablein a class should not be exposed (i.e., made public).

Making implementation details public creates interdependencies between the class and its users, subjecting the users to changes in implementation. Therefore, access should only occur via public interface methods. This makes the implementation more robust, because all data can be validated when assigned new values or the changes can be logged.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
Public Interface Design
Design Tenet: Accommodate Heterogeneity
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1021.1]

Do instance variables have public access or are they more accessible than necessary?

Procedure:

Check that the instance variable in classes does not have public access unless it is static and final.

Example:

None

2) Test: [G1021.2]

Does the class provide direct access to internal data via pass by reference?

Procedure:

Check to make sure that the methods that access the internal state do not return a reference to the internal data.

Example:

None

G1022

Statement:

Insulate public **interfaces** from compile-time dependencies.

Rationale:

There are three distinct advantages to separating interface from implementation:

- Multiple interested parties (**COIs**) can develop the interface and publish it to the user community ahead of any specific implementation. This allows groups to work independently and in parallel.
- It prevents multiple copies of the defining interface. Duplicating the code for the interface in each implementation (library, jar, and assembly) makes it difficult to maintain, especially as the interface evolves.
- It insulates developers from the constant changes in implementation.

Referenced By:

Publish and Insulate Public Interfaces
Maintainability
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Public Interface Design
Design Tenet: Accommodate Heterogeneity
Composeability

Evaluation Criteria:

1) Test: [G1022.1]

Is the packaging or deployment of the public interface self-contained and isolated to only the public interface(s)?

Procedure:

Check to make sure that the jar, library, assembly, and WSDL only contain the agreed-upon public interface (interfaces being shared externally).

Example:

None

2) Test: [G1022.2]

Does the container (jars, libraries, assemblies, WSDL) contain files other than the interface?

Procedure:

Check to make sure the library does not include or rely upon any other files such as resource files, properties files, configuration files, other libraries, XML files, and so on that would force the repackaging of the public interface.

Example:

None

3) Test: [G1022.3]

Are there any outside influences that could affect the packaging of the public interface?

Procedure:

Check the public interface for dependence on resource files, properties files, configuration files, XML files, and other libraries or packages.

Example:

None

G1027

Statement:

Internally document all source code developed with DoD funding.

Rationale:

Well-documented source code is easier to maintain and enhance over time. It is hard enough to get documentation about software and to keep it up to date. If the documentation is not internal to the source code, the chances that the software is current and up-to-date decreases. In recent years, the trend has been to generate external documentation about the software by processing the source code and comments (e.g., **Javadoc**).

In addition to documenting the functionality of the source code, it is important to capture the configuration control information (e.g., CVS).

Referenced By:

Standard Interface Documentation
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1027.1]

Do all the source code files have a header that includes a statement protecting government rights to the source code and the right to change the source code?

Procedure:

Scan each file and make sure the header includes a statement that protects the government's right to use, modify, and share the information with other government departments and agencies.

Example:

None

2) Test: [G1027.2]

Do all the source code files have a header that includes configuration information?

Procedure:

Scan each file and make sure the header also includes configuration management information such as author, date created, and a history of modifications and versions.

Example:

None

3) Test: [G1027.3]

Do all the source code files have internal documentation for attributes, methods that a computer process?

Procedure:

Scan the source files and make sure they are internally documented with tags such as Javadoc or XML tags.

Example:

None

G1030

Statement:

Use a standard GUI **component** library.

Rationale:

A predefined component library helps control cost and configuration. Licensing issues can be resolved before development begins, and component costs are minimized by avoiding library overlap.

Now that component architecture is standard, it is possible to put together applications using a variety of components from multiple vendors. These components are bundled in third-party toolkits that vastly extend the range of options available in standard Windows or Java GUI toolkits. These toolkits are in common use and possess a wide variety of pre-built components. Almost all support common **look-and-feel** (e.g., Windows or Java).

Referenced By:

Design Tenet: Accommodate Heterogeneity
Thick Clients
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1030.1]

Does the user interface code use any other toolkits besides a Standard GUI Toolkit?

Procedure:

Check to make sure the thick-client code is developed using the Swing/AWT library in Java, and the standard, included Windows Toolkit In .NET.

Example:

None

G1032

Statement:

Validate all input fields.

Rationale:

Detect errors as close to point-of-data-entry as possible. This greatly enhances the end-user experience and reduces frustration. This can be done by reducing the number of freeform text fields and using selection mechanisms such as radio buttons, option boxes, pull down lists, maps, calendars, clocks, slider bars, and other numeric validation entries.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
Design Tenet: Accommodate Heterogeneity
Design Tenet: Enterprise Service Management
Presentation Tier
Human-Computer Interaction
Design Tenet: Open Architecture
Validate Input

Evaluation Criteria:

1) Test: [G1032.1]

Do the GUI screens use non-freeform text entry fields?

Procedure:

Scan the GUI code looking for the use of non-freeform text data entry mechanisms.

Example:

None.

G1043

Statement:

Separate formatting from data through the use of **style sheets** instead of hard coded **HTML** attributes.

Rationale:

Formatting information will be located in one location instead of scattered throughout each individual Web page of a Web site. This makes a Web site more maintainable.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Style Sheets
Design Tenet: Open Architecture
Maintainability
Browser-Based Clients
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1043.1]

Are any formatting attributes used in any of the HTML tags?

Procedure:

Search all Web pages and make sure there are no formatting attributes such as align, color, font, or size in any tags.

Example:

None

G1044

Statement:

Comply with Federal accessibility standards contained in Section 508 of the Rehabilitation Act of 1973 (as amended) when developing software user interfaces.

Rationale:

Applicable software must comply with Federal standards to enable better application use for those with disabilities.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
Designing User Interfaces for Accessibility

Evaluation Criteria:

1) Test: [G1044.1]

Do all Web document **HTML**, **JSP**, **ASP**, and **CSS** follow the Disability Act guidelines?

Procedure:

Check to make sure all Web documents follow the guidelines.

Use available validation tools to validate Section 508 accessibility and WAI accessibility. Go to http://www.contentquality.com/Default.asp to validate the page.

Example:

None

G1045

Statement:

Define **XML** format information separately in **XSL**.

Rationale:

XML documents should be free of any presentation information and should only contain data. Separating presentation data from content allows multiple presentations for the same content data.

Referenced By:

Defining XML Schemas
Design Tenet: Service-Oriented Architecture (SOA)
XML Rendering
Reusability
Design Tenet: Accommodate Heterogeneity
Composeability
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1045.1]

Check for presentation information in XML documents?

Procedure:

Does the XML document contain only data?

If the XML document is not an document, does it contain presentation information?

Example:

None

G1050

Statement:

In **ASP**, isolate the presentation tier from the middle tier using **COM** objects.

Rationale:

This is the best way to isolate the presentation tier from the middle tier in ASP.

Referenced By:

Active Server Pages (ASP)
Design Tenet: Service-Oriented Architecture (SOA)
Composeability
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1050.1]

Is all the middle tier code isolated from the presentation tier in ASP via COM?

Procedure:

Verify that ASP files do not contain middle-tier code. Instead, this code should be in COM objects referenced from the ASP.

Example:

None

G1052

Statement:

Use the code-behind feature in ASP.NET to separate presentation code from the business logic.

Rationale:

Separating presentation code from business logic allows the developers and content designers to work independently. It also makes the code more maintainable because changes in the design elements or business elements do not affect each other.

Referenced By:

Design Tenet: Open Architecture
Composeability
Design Tenet: Service-Oriented Architecture (SOA)
Active Server Pages for .NET (ASP.NET)
Maintainability

Evaluation Criteria:

1) Test: [G1052.1]

Is there code in ASP pages?

Procedure:

Check to make sure that ASP files have the code-behind attribute in the first line instead of embedded C# code in the ASP.

Example:

None

G1053

Statement:

Do not embed HTML code in any code-behind code used by aspx pages.

Rationale:

Intermixing VB or C# or C++ with presentation code (HTML) makes the code unnecessarily difficult to maintain by both the developer and designer. This is similar in concept to Java's not embedding HTML code in **servlets**.

Referenced By:

Active Server Pages for .NET (ASP.NET)
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Maintainability

Evaluation Criteria:

1) Test: [G1053.1]

Check for HTML code in code-behind code.

Procedure:

Check the code-behind file (**`.aspx.vb`** for example) for any HTML tags.

Example:

None

G1056

Statement:

Specify a versioning policy for **.NET** assemblies.

Rationale:

Versioning assemblies and configuring dependent assemblies allow the **Common Language Runtime** (**CLR**) to load the proper assemblies at runtime for an application. This insulates the application from system configuration changes.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
Active Server Pages for .NET (ASP.NET)
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1056.1]

Does the application assembly have versioning information?

Procedure:

Check the application assembly manifest for versioning information.

Use the .NET configuration tool to check for versioning policy and versioning information.

Example:

None

G1058

Statement:

Use the Model, View, Controller (MVC) pattern to decouple presentation code from other tiers.

Rationale:

Separating data-layer code from presentation-layer code provides the ability to base multiple views on the same model. This is especially important in the enterprise model because often, the user interface varies with the device (browser, mobile phone, thick client, etc.).

Isolating different layers allows changes to occur in each layer without impacting other layers. For instance, if the data layer (model) decides to switch databases, the changes are isolated to the data layer and do not affect the view layer or controller layer.

Lastly, because MVC architecture enforces separation between presentation, processing, and data layer, this allows functionality to be loosely coupled and therefore more suited for reuse.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Reusability
Active Server Pages for .NET (ASP.NET)
Design Tenet: Accommodate Heterogeneity
Composeability
Active Server Pages (ASP)
Java Server Pages (JSP)
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1058.1]

Does the application use a Model 2 (MVC) pattern?

Procedure:

Check to see if all requests are being mapped to a single controller servlet.

Check that all page rendering are being done by a and not a .

Example:

None

2) Test: [G1058.2]

Does the application enforce clear separation between data layer (model), presentation layer (view), and middle/ business layer (controller)?

Procedure:

Check to make sure the application presentation is not accessing the database directly.

Check to make sure the application data layer (model) is not implementing business logic (store procedures).

Check to make sure the middle/business layer (controller) does not contain presentation code. For example, make sure servlets do not generate HTML.

Make sure access to the database is isolated to Data Access Object instead of proliferated throughout the middle layer.

Example:

None

G1060

Statement:

Encapsulate Java code that is used in **JSP**(s) in tag libraries.

Rationale:

Separating code from presentation allows developers and designers to work independently. It makes the code reusable and more maintainable because it is defined in a tag library.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Composeability
Java Server Pages (JSP)
Maintainability
Reusability

Evaluation Criteria:

1) Test: [G1060.1]

Do the JSP pages use tag libraries?

Procedure:

Look through the JSP pages for embedded Java source code.

Example:

None

G1071

Statement:

Use vendor-neutral interface connections to the enterprise (e.g., **LDAP**, **JNDI**, **JMS**, databases).

Rationale:

Increase **portability** and maintainability. Many of the newer connection mechanisms are vendor-neutral. Use these instead of isolation design patterns or vendor-specific connection mechanisms.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Maintainability
Design Tenet: Open Architecture
Interoperability
JNDI Security
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1071.1]

Is the connection mechanism vendor-neutral?

Procedure:

Examine the source code for vendor-specific imports or includes. Use only standard APIs.

Example:

None

G1073

Statement:

Isolate vendor extensions to enterprise-services standard interfaces.

Rationale:

Vendor extensions are convenient but help create "vendor lock" and reduce vendor neutrality and migration. It is best to avoid these extensions altogether. If that is not possible, then isolate them in an **adapter** or a wrapper-like construct.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Interoperability
Maintainability
Publish and Insulate Public Interfaces

Evaluation Criteria:

1) Test: [G1073.1]

Are vendor extensions to enterprise services used?

Procedure:

Make sure that no vendor-specific code is included or imported except as part of an adapter or wrapper.

Example:

None

G1078

Statement:

Document the use of non-**Java EE**-defined **deployment descriptors**.

Rationale:

Deployment descriptors that are not defined by the J2EE specification are not portable between **application servers**. For example, BEA WebLogic has a vendor-specific deployment descriptor called `weblogic-ejb-jar.xml` and JBoss has a vendor specific deployment descriptor called `jboss-jar.xml` .

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Design Tenet: Open Architecture
Java EE Environment

Evaluation Criteria:

1) Test: [G1078.1]

Are all the XML files that are not part of the Java EE specification identified in a delivered document?

Procedure:

Search all XML documents in the META-INF and WEB-INF directories and identify any XML files that are not defined by Java EE. These files should be in a README or other delivered file that describes their purpose:

Example:

Web application	`WEB-INF/web.xml`
EJB JAR	`META-INF/ejb-jar.xml`
J2EE Connector	`META-INF/ra.xml`
Client application	`META-INF/application-client.xml`
Enterprise application	`META-INF/application.xml`

G1079

Statement:

Isolate tailorable data values into the **deployment descriptors** for **Java EE** applications.

Rationale:

Do not hard-code tailorable data into source files. The standard location for tailorable data for Java EE applications is in deployment descriptors. Developers should not "reinvent the wheel" by creating a non-standard mechanism for retrieving configurable data. Make tailorable data accessible through application contexts provided by the application **container** (**Java EE application server**).

Referenced By:

Java EE Environment
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
JNDI Security

Evaluation Criteria:

1) Test: [G1079.1]

Is tailorable data configured using deployment descriptors?

Procedure:

Check the deployment descriptor for instances of tailorable data.

Example:

Name-value pairs such as **environment variables** configured using resource-env-ref elements.

JNDI locations configured using resource-ref elments.

G1080

Statement:

Adhere to the **Web Services Interoperability Organization** (**WS-I**) Basic Profile specification for **Web service** environments.

Rationale:

Most of the **COTS** Web service products have already met this requirement. This is intended to cause a rejection of the non-standard Web server.

The WS-I Basic Profile specification is available from the Web Services Interoperability Organization Web site: WS-I Org Basic Profile.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Web Services Compliance
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1080.1]

Is the Web service product WS-I Basic Profile specification compliant?

Procedure:

Identify the Web service product being used, and verify through a literature search that it is WS-I Basic Profile specification compliant.

Example:

None

G1082

Statement:

Use the document-literal style for all data transferred using **SOAP** where the document uses the **World Wide Web Consortium** (**W3C**) **Document Object Model** (**DOM**).

Rationale:

The document-literal style requires defining the input and output parameters to a Web service as documents that follow the W3C Document Object Model (DOM). The DOM acts as a contract between the **producer** and the **consumer** of the Web service that is formal, well-defined, and rigorous. Validating the DOM against an **XML** Schema Definition (**XSD**) can help resolve discrepancies in the interface.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Maintainability
Web Services Compliance
SOAP
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Scalability

Evaluation Criteria:

1) Test: [G1082.1]

Does the **WSDL** define input, output, or returned parameters as Documents that follow the **W3C** Document Object Model (**DOM**)?

Procedure:

Review all WSDL files used to describe a Web service, and make sure they only pass documents. Document types should be **`xsd:anyType`**.

Example:

None

G1083

Statement:

Do not pass **Web Services-Interoperability Organization** (**WS-I**) **Document Object Model** (**DOM**) documents as strings.

Rationale:

Because of the relative simplicity of converting an **XML** document to a string, it is easy to pass an entire document as a string rather than as an XML document. This can cause problems if the document contains tags that are similar to the tags used in the **SOAP**. Passing it as an XML document ensures that the document is treated as a single entity.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Design Tenet: Open Architecture
Web Services Compliance
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability

Evaluation Criteria:

1) Test: [G1083.1]

Does the **WSDL** define input, output, or returned parameters as strings?

Procedure:

Review all the WSDL files used to describe a Web service and make sure that they only pass documents, not strings. Document types should be **`xsd:anyType`**.

Example:

None

G1084

Statement:

Validate documents transferred using **SOAP** against the **W3C XML** Standard by an **XML Schema Definition** (**XSD**) defined by the **Community of Interest** (**COI**).

Rationale:

Numerous **COIs** are defining data specific to their needs. Many are capturing the data exchange requirements through **XML schemas**. **COI** information service definitions identify the appropriate schema. **SOAP** Web service implementations per the **COI** should be faithful to these requirements. Use of **COI** schemas will minimize the risk to interoperability.

For example, the Joint Air and Missile Defense (JAMD) **COI** is working in accordance with the DoD Network Centric Data Strategy.

Referenced By:

SOAP
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
WSDL

Evaluation Criteria:

1) Test: [G1084.1]

Has the Program adopted **COI** (Community of Interest) data schemas?

Procedure:

Check the DoD Metadata Registry for the **COI** schemas to compare to program **WSDL** references. Check code for validation processing.

Example:

None

G1085

Statement:

Establish a **registered namespace** in the **XML Gallery** in the **DoD Metadata Registry** for all DoD Programs.

Rationale:

A **registered namespace** permits unique identification and categorization of a Program which avoids name collisions and conflicts. The DoD Net-Centric Data Strategy requires storing data products in shared spaces to provide access to all authorized users and tagging these data products with **metadata** to enable discovery of data by authorized users. The use of a unique **registered namespace** provides an absolute identifier to products associated with a particular product and is an **XSD** schema requirement.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
WSDL
Using XML Namespaces
Interoperability

Evaluation Criteria:

1) Test: [G1085.1]

Does the Program have an assigned namespace in the **DoD Metadata Registry**?

Procedure:

Check the **DoD Metadata Registry** to determine whether program is associated with **COI**(s).

Example:

None

G1087

Statement:

Validate all **Web Services Definition Language** (**WSDL**) files that describe **Web services**.

Rationale:

Manually editing a **WSDL** file is error-prone, work-intensive, and hard to maintain. However, if the user wants to do it, there is no way to detect a manually edited file from one that was auto generated. The important thing is not how the **WSDL** file is generated but rather that the **WSDL** file is valid. It must be validated with a **WSDL** validator.

Note: Not all **WSDL** files that are generated and valid are necessarily interoperable.

Referenced By:

Web Services
WSDL
Design Tenet: Accommodate Heterogeneity
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Insulation and Structure

Evaluation Criteria:

1) Test: [G1087.1]

Can the **WSDL** file be validated?

Procedure:

Download a validation tool and test WSDL files.

Example:

Sample tools:

WS-I Organization:	http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools
Eclipse:	http://dev.eclipse.org/viewcvs/indextech.cgi/wsvt-home/main.html?rev=1.20
XMethods:	http://xmethods.net/ve2/Tools.po
Pocket Soap:	http://pocketsoap.com/wsdl/

G1088

Statement:

Use isolation design patterns to define system functionality that manipulates **Web services**.

Rationale:

Insulating **SOAP** Web-service manipulation using standard abstraction patterns such as a **proxy** or **adapter** insulates the software system from changes in the Web service interface and promotes maintainability.

Referenced By:

Web Services
SOAP
Design Tenet: Scalability
Design Tenet: Accommodate Heterogeneity
Insulation and Structure
Maintainability
Design Tenet: Open Architecture
Composeability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1088.2]

Are Web service calls isolated in a single adapter or proxy object?

Procedure:

Check to see if all Web service calls are isolated to a single adapter or proxy object.

Example:

None

2) Test: [G1088.1]

Are Web service calls inside of the application code?

Procedure:

Check for proliferation of Web service calls inside an application.

Example:

None

3) Test: [G1088.3]

Are SOAP-client calls inside the application code?

Procedure:

Check to see if SOAP-client code is proliferated inside the application code?

Example:

None

G1090

Statement:

Do not **hard-code** a **Web service's endpoint**.

Rationale:

This causes unnecessary dependencies between the client code and the Web service that it uses.

Sometimes hard-coding may be unavoidable. For example, many tools provided by Web service vendors hard-code the Web service's URL in the generated client-side helper classes.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Web Services
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1090.1]

Are there any hard-coded URLs in the client-side code?

Procedure:

Parse the client code looking for hard-coded URLs.

Example:

The Java code samples below illustrate how this might be done. The first sample shows parameters that are hard-coded; the second sample shows how parameters and Web service endpoints are insulated.

1. Hard-coded parameters:

```
// Sample code that has hard-coded parameters
// before applying insulation
public static void main
  ( String[] args
  ) throws Exception
{ //The SOAP endpoint
  String sSoapEndpoint
     = "http://live.capescience.com:80"
       + "/ccx/AirportWeather";
   AirportWeatherClient myProxy = null;
  try
  { myProxy
     = AirportWeatherClientFactory.create
        ( sSoapEndpoint);
   System.out.println
     ("Location: "
      + myProxy.getLocation(args[0])
     );
   //rest of code removed for brevity
  } // End try
  Catch ( Exception exception )
  { System.out.println("Error: " + exception);
  } // End catch
};//end of main program
```

2. Insulated parameters and Web service endpoints

a. Property file - this code shows the property file itself:

c. Client sample code:

```
import java.io.*;
import java.rmi.*;
import java.util.*;
import AirportWeatherClient; // auto-generated SOAP
                             // client from IDE */
public class WeatherProxy
  implements airportWeatherProxy
{
 //
 //code removed for brevity
 //
 public WeatherProxy
  ( String propFileStr )
 { try
   { getEndPoint(propFileStr);
   } // End try
   catch(Exception e)
   { // Handle exception here
   } // End catch
   connect2SOAP();
 }// End constructor
 /* public api's */
 public String getLocation()
 { return location;
 } // End getLocation
   . . . // Other public API's removed for brevity
 private void getEndPoint
  ( String propsFile )
  throws Exception
 { if ( propsFile == null || propsFile.length() == 0 )
   { throw new Exception
       ( "SOAP EndPoint parameter not defined");
   } // End if
   props = new Properties();
   try
   { InputStream is = new FileInputStream(propsFile);
     props.load(is);
     is.close();
    } // End try
    catch ( Exception exception )
    { throw new Exception
        ( "can't read props file " + propsFile);
    } // End catch
    Enumeration enum = props.propertyNames();
    while ( enum.hasMoreElements() )
    { String endPointString = null;
      String propName = enum.nextElement().toString();
      if ( propName.equals ( endPointString ) )
     { soapEndpoint = props.getProperty( propName );
       break;
     } // end if
    } // End while
  }//end getEndPoint
 private void connect2SOAP()
 { try
   { myProxy
       = AirportWeatherClientFactory.create
           ( soapEndpoint );
     . . . //code removed for brevity
   } // End try
   catch ( Exception exception )
   { System.out.println
```

```
       ( "Error connecting to SOAP server: "
          + exception
       );
   } // End catch
 } // End connect2SOAP
  private Properties props = null;
  private String propsFile = null;
  private AirportWeatherClient myProxy = null;
  private String soapEndpoint = null;
  private String location = null;
}//end WeatherProxy
public class Weather
{ private static WeatherProxy myWeatherProxy = null;
  public static void main
    ( String[] args
    ) throws Exception
  { try
    { myWeatherProxy = new WeatherProxy ( args[0] );
    } // End try
    Catch ( Exception exception )
    { throw new Exception
        ( "can't connect to SOAP server");
    } // End catch
    System.out.println
      ( "Location: "
        + myWeatherProxy.getLocation()
      );
    . . . //code deleted for brevity
  }//end main
}//end Weather
```

G1093

Statement:

Implement exception handlers for **SOAP**-based **Web services**.

Rationale:

SOAP exceptions result when there are connectivity problems or violations in the SOAP protocol between the client and the server.

Referenced By:

Interoperability
Error Handling
Design Tenet: Accommodate Heterogeneity
SOAP
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Enterprise Service Management
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1093.1]

Does the Web application client have exception handlers for **`SOAPExceptions`**.

Procedure:

Check to see that the Web application client has an exception block specifically for **`SOAPException`**.

Example:

None

2) Test: [G1093.2]

Does the Web application client test the SOAP response for a fault?

Procedure:

Verify the Web application client handles a true value returned from the **`response.generatedFault`**.

Example:

None

G1094

Statement:

Catch all exceptions for application code exposed as a **Web service**.

Rationale:

Any exception can reveal system internals and thus compromise security. Also, internal exceptions are not user friendly.

Referenced By:

Maintainability
Error Handling
Design Tenet: Enterprise Service Management
Handle Exceptions

Evaluation Criteria:

1) Test: [G1094.2]

Does each exposed Web method catch all possible runtime exceptions and re-throw a declared application runtime exception?

Procedure:

Verify that each exposed Web method has an exception block that catches all possible exceptions and then re-throws them as a declared application exceptions.

Example:

None

2) Test: [G1094.1]

Does each exposed Web method catch all possible exceptions and re-throw a declared application exception?

Procedure:

Verify that each exposed Web method has an exception block that catches all possible exceptions and then re-throws them as a declared application exceptions.

Example:

None

G1095

Statement:

Use **W3C** fault codes for all **SOAP** faults.

Rationale:

Having predefined and accepted fault codes allows consumers to handle SOAP faults appropriately without prior knowledge of custom fault codes.

Referenced By:

SOAP
Design Tenet: Open Architecture
Error Handling
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Maintainability
Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test: [G1095.1]

Does the Web application throw fault codes from the accepted list of fault codes?

Procedure:

Verify that each fault code thrown by the Web application is from the accepted list of SOAP fault codes defined by the W3C.

Example:

None

G1101

Statement:

Use **Web services** to bridge **Java EE** and **.NET**.

Rationale:

The easiest and best way to bridge Java EE and .NET is to define a Web service.

There are other ways to bridge Java EE and .NET using **COTS** products. If used, these should follow the **ANSI** Abstract Syntax Notation One (ASN.1) standard (http://asn1.elibel.tm.fr/en/standards/index.htm#asn1).

ASN.1 is a formal notation for describing data transmitted by telecommunications protocols. It applies regardless of language implementation, physical representation of this data, application, and degree of complexity (http://asn1.elibel.tm.fr/en/introduction/index.htm).

Referenced By:

.NET Framework
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Interoperability

Evaluation Criteria:

1) Test: [G1101.1]

Are Java and .NET files in the project?

Procedure:

Look for files with the .java, .class, .obj, .cs, .cc, or .c extensions existing with the source code.

Example:

None

G1118

Statement:

Localize **CORBA** vendor-specific source code into separate **modules**.

Rationale:

The general guidance is to minimize CORBA vendor-specific source code, while recognizing that vendor-specific features are necessary in certain circumstances. However, isolating vendor-specific code reduces maintenance effort.

Vendor capabilities tend to change more rapidly than CORBA-standard specifications. Experience shows that vendor updates frequently require modification to application source code, due to changing vendor interface conventions. These modifications impose vendor-version-specific constraints on the application, thereby complicating maintenance.

Example

Encapsulating CORBA ORB operations

The following examples show how to encapsulate binding operations for a C++ ORB, and naming service operations for a Java ORB.

C++ ORB binder template

The code below shows a sample template for binding to the C++ ORB. IONA's ORBIX was used in this example.

```
/* ====================================================
ServerBinder.h (Template)
this is a generic binder to ORBIX
==================================================== */
#ifndef _BINDER_H_
#define _BINDER_H_
#ifndef IOSTREAM_H
#define IOSTREAM_H
#include <iostream.h>
#endif
#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
template <class SERVERNAME, class VARPTR>
class Binder
{ private:
    char* serverName;
  public:
    Binder(char* svName):serverName(svName){};
    ~Binder(){};
    int bind( VARPTR* p)
    { int attempts = 0, success = 0;
      int maxtries = 5, retval = 0;
      while ( ( attempts < maxtries )
             && (!success)
            )
      { ++attempts;
        cout << "Binding to server, attempt "
            << attempts
            << endl;
        try
        { (*p) = SERVERNAME::_bind();
```

```
           cout << "Bound to server"
               << endl;
           success = retval = 1;
         } // End try
         catch ( CORBA::SystemException &systemException )
         { cout << "SystemException, ServerBinder::bind"
               << endl
               << systemException;
           success = 1;
           retval = 0;
         } // End catch SystemException
          catch (...)
          { cout << "unknown Exception, ServerBinder::bind"
                << endl;
            success = 1;
            retval = 0;
          } // End catch all
       } //end while
       return retval;
     } //end bind
} //end Binder
#endif
```

Ada ORB binder template for C++

The code below shows a C++ template for binding to an Ada ORB. ORBexpress was used in this example.

```
/* ====================================================
ada_binder.h (Template)
this is a generic binder to ORBExpress
==================================================== */
#ifndef _ADA_BINDER_H_
#define _ADA_BINDER_H_
#ifndef IOSTREAM_H
#define IOSTREAM_H
#include <iostream.h>
#endif
#ifndef STDLIB_H
#define STDLIB_H
#include <stdlib.h>
#endif
template <class SERVERNAME, class VARPTR >
class Ada_Binder
{ private:
    char* adaIorString;
  public:
     Ada_Binder
       ( char* iorString)
       : adaIorString ( iorString )
     {};
     ~Ada_Binder(){};
     int bindToAda( VARPTR* p)
     { int attempts = 0, success = 0;
       int maxtries = 5, retval = 0;
       while ( ( attempts < maxtries)
              && (!success)
             )
       { ++attempts;
         cout << "Binding to server, attempt "
             << attempts
             << endl;
         try
         { cout <<"adaIorString:"
               << endl
               << adaIorString
               << endl;
               (*p) = SERVERNAME::_bind(adaIorString);
//can't use string_to_object in this version
//it kills the ada IOR
//            CORBA::Object_ptr myptr
              CORBA::Orbix.string_to_object
                ( adaIorString );
//            (*p) = SERVERNAME::_narrow(myptr);
            cout << "Bound to server" << endl;
            success = retval = 1;
         } // End try
         catch (CORBA::SystemException& systemException)
         { cout << "SystemException, "
               << "AdaServerBinder::bind"
               << endl
               << systemException;
           success = 1;
           retval = 0;
         } // End SystemException
         catch (...)
         { cout << "Unknown Exception, "
               << "AdaServerBinder::bind"
               << endl;
           success = 1;
           retval = 0;
         } // End catch all
      } // end while
      return retval;
    } // end bind
} // end ADA_Binder
#endif
```

Example

Naming service operations for a Java ORB

Java helper class

This example is a helper class, JavaNamingHelper.java, that encapsulates CORBA naming service operations for all services to use. We used Java JDK 1.4 ORB to create this example.

```
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
import CBRNSensors.JSLSCAD.*;
public class JavaNamingHelper
{ static NamingContext nameSvc = null;
  static org.omg.CORBA.Object objref = null;
  static JSLSCADSensor myCBRNSensor = null;
  static org.omg.CORBA.Object myobj = null;
  public JavaNamingHelper()
  {
  }
  private static void showNamingContext
    ( org.omg.CORBA.ORB myorb )
  {
  public static NamingContext getNamingSvc
    ( org.omg.CORBA.ORB lclorb,
      String nameSvcName
    )
  { NamingContext lclNameSvc = null;
    try
    { org.omg.CORBA.Object nameSvcObj
        = lclorb.resolve_initial_references
            ( "NameService" );
       // . . . other business logic removed
       //       for brevity
    } // End try
    catch(org.omg.CORBA.COMM_FAILURE cf)
    { . . . // error code goes here
    } // End cstch
    catch ( org.omg.CORBA.ORBPackage.InvalidName invalidName)
    { . . . // error code goes here
    } // End catch
    catch ( SystemException systemException )
    { . . .// error code goes here
    }
  } // End getNamingSvc
  public static org.omg.CORBA.Object getObjFromNameSvc
    ( org.omg.CORBA.ORB myorb,
      String targetSensorName
    )
  { . . . // business logic goes here
  } //end getObjFromNameSvc
  public static int setObj2NameSvc
    ( org.omg.CORBA.ORB myorb,
      BasesSensor mySensor,
      String targetSensorName
    )
  {. . . // business logic goes here
  }//end setObj2NameSvc
}; //end class JavaNamingHelper
```

Java server implementation

The code below is a sample Java server implementation that uses the naming service helper class.

```
import java.io.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import org.omg.CORBA_2_3.ORB.*;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContext.*;
import org.omg.CosNaming.NamingContextPackage.*;
class MyServer
{ public static Properties props;
  public static ORB myorb = null;
  public static NamingContext nameSvc = null;
  public static RootSensor mySensor = null;
  public static String propertyFilePath = null;
  public static final String MY_SENSOR_NAME = "MYSENSOR";
  static public void main(String[] args)
  { // handle arguments
    System.out.println(" CORBA Server starting...\n");
    try
    { // Initialize the ORB.
      myorb = ORB.init(args, props);
      //instantiate servant and create ref
      POA rootPOA
        = POAHelper.narrow(myorb.resolve_initial_references
           ( "RootPOA" );
     . . . // rest of initialization code goes here
    } // End try
    catch ( org.omg.CORBA.ORBPackage.InvalidName invalidName )
    { . . . //error code goes here
    } // End invalidName
    // other exception types to catch go here
    catch ( SystemException systemException)
    {  System.err.println ( systemException );
    } // End systemException
    // naming service hookup
    JavaNamingHelper.setObj2NameSvc
      ( myorb,mySensor,
        MY_SENSOR_NAME
      );
    try
    { System.out.println(" Ready to service requests\n");
      myorb.run();
    } // End try
    catch(SystemException systemException)
    { System.err.println ( systemException );
    } // End catch systemException
  } // End static block
} // End MyServer
```

Java client implementation

The code below is a sample client implementation that uses the naming service helper class.

Referenced By:

Design Tenet: Open Architecture
CORBA
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1118.2]

Are any non-CORBA compliant CORBA:: objects declared or defined in the module?

Procedure:

Review the code for a service that can be used to obtain configuration.

Example:

None

2) Test: [G1118.1]

Does the module contain vendor names anywhere in code text?

Procedure:

Review the code looking for a service that can be used to obtain configuration.

Example:

None

G1119

Statement:

Isolate user-modifiable configuration parameters from the **CORBA** application source code.

Rationale:

Configuration parameters control the behavior of the CORBA **ORB** service environment and client/service processes during startup, execution, and termination. This parameterization allows execution-time control modification without having to rebuild, reinstall, or redeploy.

Configuration defines the state of the client-and-service environment throughout the lifetime of the processes involved. This relates to considerations such as the allocation of threading and resources, **POA** policies, the instantiation of servants and their invocations, failure and security behavior, connection management, quality of service prioritization, and so forth. The point is that CORBA provides an extremely complex but flexible environment for distributed computing interaction. Consequently, the designer requires flexible guidance to handle this option-rich environment.

Configuration processes and their related parameters fall into two categories. The first involves configuration matters, which are defined to be perpetually static by the system architecture. The second involves matters that are intended to be modifiable by users.

The first category, immutable configuration settings, relates to fundamental underlying assumptions that are foundational for the implementation. These are matters for which no user modification is ever intended as it would lead to unspecified behavior. Consider the example of a service implementation that is programmed to be single threaded. In this case, multi-threading controls are irrelevant and multiple instantiation would lead to dangerous confusion. For immutable configuration parameters, localized and well-commented implementation in the application source code is appropriate.

For user-modifiable configuration settings, there are two further by-design divisions. The first involves configuration settings that are intended to be accessible by distributed processes. The second involves host-specific settings which relate to resources locally available, for which remote access is not desired. These are discussed in the related sublevel guidance

Referenced By:

CORBA
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1119.1]

See G1204.

Procedure:

Example:

2) Test: [G1119.2]

See G1205 .

Procedure:

Example:

G1121

Statement:

Do not modify **CORBA** Interface Definition Language (**IDL**) compiler auto-generated stubs and skeletons.

Rationale:

The purpose of the IDL auto-generated stub and skeleton files is to provide a source code facility/mechanism for the developer in a specific language to use the IDL-described object interface in that specific language. The internal content of these files changes with the application's IDL modification, with IDL compiler-environment configuration settings, and with vendor-product compiler and **ORB** upgrades. By design, these files are not intended to be modified by the application developer. Developer modification of any auto-generated stub or skeleton file will typically lead to very severe maintenance hazards and failed application rebuild results.

The stub files describe the language source-code interface from the client side. Their use involves including the client stub header in the application's call invocation code.

The skeleton files describe the language source code interface from the service implementation side. Their use involves including the skeleton header in the application's operator implementation code. Their use also requires developer modification of a renamed clone of the auto-generated skeleton body file. These techniques are described in every ORB vendor's programming reference manuals.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
CORBA

Evaluation Criteria:

1) Test: [G1121.1]

Is any application code contained in the auto-generated code?

Procedure:

Inspect the auto-generated file creation/modification dates to verify that no tampering occurred after the IDL compilation step in the build process.

Example:

The following examples are all based upon a single CORBA IDL interface.

MyIdlInterface.idl

```
interface MyIdlInterface
{
  readonly attribute string version;
  void stop();
  void start();
  string error();
}; // End MyIdlInterface
```

ORBExpress compiler

The ORBExpress IDL compiler generates these files:

- **`myIdlInterface.h`** - Client-side stub header
- **`myIdlInterface.cxx`** - Client-side stub implementation
- **`MyIdlInterface_s.h`** - Abstract servant header
- **`MyIdlInterface_s.cxx`** - Abstract servant implementation
- **`MyIdlInterface_impl.h`** - Server implementation header
- **`MyIdlInterface_impl.cxx`** - Server implementation implementation

Note: *The only files that should be edited are **`MyIdlInterface_impl.h`** and **`MyIdlInterface_impl.cxx`** . The IDL compiler checks for the existence of the implementation (i.e. _impl) files and will not overwrite them.*

MyIdlInterface_impl.cxx

```
// Generated for interface MyIdlInterface
// in myIdlInterface.idl
#include "MyIdlInterface_impl.h"
MyIdlInterface_impl::MyIdlInterface_impl
  ( PortableServer::POA* oe_poa,
    const char* oe_object_id
  ) : POA_MyIdlInterface
        ( oe_object_id,
          oe_poa
        )
{ . . . // TO DO: add implementation code here
} // emd constructor
MyIdlInterface_impl::MyIdlInterface_impl
  ( const MyIdlInterface_impl& obj )
  : POA_MyIdlInterface(obj)
{ . . . // TO DO: add implementation code here
} // End constructor
MyIdlInterface_impl::~MyIdlInterface_impl()
{ . . . // TO DO: add implementation code here
} // End destructor
CORBA::Char* MyIdlInterface_impl::version
  ( CORBA::Environment& _env )
{ return CORBA::string_dup(_version);
} // End version
void MyIdlInterface_impl::stop
   ( CORBA::Environment& _env )
{ . . . // TO DO: add implementation code here
} // End stop
void MyIdlInterface_impl::start
   ( CORBA::Environment& _env )
{ . . . // TO DO: add implementation code here
} // End start
CORBA::Char* MyIdlInterface_impl::error
  ( CORBA::Environment& _env )
{ CORBA::Char* result;
  . . . // TO DO: add implementation code here
  return result;
} // End error
```

Java JDK compiler

The Java JDK IDL compiler generates these files:

- **`MyIdlInterface.java`**
- **`MyIdlInterfaceHelper.java`**

- **`MyIdlInterfaceHolder.java`**
- **`MyIdlInterfaceOperations.java`**
- **`MyIdlInterfacePOA.java`**
- **`_MyIdlInterfaceStub.java`**

MyIdlInterfacePOA.java

```
/**
 * MyIdlInterfacePOA.java .
 * Generated by the IDL-to-Java compiler
 * (portable), version "3.1"
 * from myIdlInterface.idl
 */
public abstract class MyIdlInterfacePOA
  extends org.omg.PortableServer.Servant
  implements MyIdlInterfaceOperations,
             org.omg.CORBA.portable.InvokeHandler
{ . . . // rest of the auto-generated code removed for brevity
} // End MyIdlInterfacePOA
```

MyIdlInterfaceImpl.java

```
package myIdlImpl;
```

G1123

Statement:

Use the Fat Operation Technique in **IDL** operator invocation.

Rationale:

This reduces the CORBA messaging overhead. The performance cost of network CORBA messaging is determined by two factors: latency and marshaling rate. Call latency is the minimum cost of sending any message at all. The marshaling rate is determined by the sizes of sending and receiving parameters and of return values.

In the situation of a large number of objects involving objects that hold a small amount of stat, the call latency cost far exceeds the marshalling costs. Taking advantage of this reality, the "Fat Operation Technique" involves constructing structure objects which hold an aggregation of related attributes, and using the resulting structures in operation invocation parameters and returns. This amounts to transferring a larger amount of information with each network transaction.

For more information, see "Advanced CORBA Programming with C++" by Henning & Vinoski, 1999 Addison Wesley, Chapter 22.

Referenced By:

CORBA
Design Tenet: Scalability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1123.1]

Does the IDL contain function calls which have structure objects that are passed as parameters or returned from operators?

Procedure:

Inspect the IDL file and manually check for parameters or returns using objects defined as structures, and verify that they are passed from methods also declared in the IDL.

Example:

None

G1125

Statement:

Use the **Department of Defense Metadata Specification** (**DDMS**) for standardized tags and taxonomies.

Rationale:

These standardized tags or Metacards will be developed, maintained, and placed under configuration as appropriate and will comply with the **DDMS** and **COI** guidance. These include specifications defining the tagging for security classification and dissemination control. See the DoD Discovery Metadata Specification Web site (http://metadata.dod.mil/mdr/irs/DDMS/) for the current **DDMS** standards.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Make Data Visible
Design Tenet: Provide Data Management
Design Tenet: Open Architecture
Metadata Registry
Design Tenet: Accommodate Heterogeneity
Interoperability

Evaluation Criteria:

1) Test: [G1125.1]

Has the Program documented the profile used for published data assets in accordance with guidance?

Procedure:

Check the DoD Metadata Registry to determine whether the program is associated with **COI**(s).

Example:

None

G1127

Statement:

Use a **UDDI** specification that supports publishing discovery services.

Rationale:

UDDI provides a registration for services, and the **OASIS** UDDI 2.0 specification has become a standard method for publishing discovery services.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Universal Description, Discovery, and Integration (UDDI)
Design Tenet: Accommodate Heterogeneity
Interoperability

Evaluation Criteria:

1) Test: [G1127.1]

Are the Web services registered in a **UDDI** registry?

Procedure:

Verify the registration in the UDDI registry.

Example:

None

2) Test: [G1127.2]

Is the registry **UDDI** 2.0 or higher?

Procedure:

Determine if the particular UDDI registry is UDDI Version 2.0 or higher.

Example:

None

G1131

Statement:

Use industry standard Universal Description, Discovery, and Integration (**UDDI**) **APIs** for all UDDI inquiries.

Rationale:

There is a standard **API** that uses **SOAP** messages to communicate with the UDDI registry. To increase compatibility and portability, use this API exclusively.

Referenced By:

Design Tenet: Open Architecture
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Universal Description, Discovery, and Integration (UDDI)
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1131.1]

Are all the interfaces to the UDDI registry made using the UDDI standard API?

Procedure:

The standard API for UDDI is SOAP based. Requests and responses are passed using documents. Test the traffic flow between the client and the UDDI registry for messages that are defined in the UDDI specification. Use standard libraries to send and receive the messages (e.g., JUDDI for Java).

Checking for the use of packages like JUDDI does not require the application to be running.

Example:

The following is an example as provided in the UDDI API reference: http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#_Toc25137712 .

find_binding

The find_binding API call returns a bindingDetail message that contains zero or more binding Template structures matching the criteria specified in the argument list.
Syntax

Syntax

Arguments

serviceKey	This uuid_key is used to specify a particular instance of a businessService element in the registered data. Only bindings in the specific businessService data identified by the serviceKey passed will be searched.

maxRows	This optional integer value allows the requesting program to limit the number of results returned.
findQualifiers	This optional collection of findQualifier elements can be used to alter the default behavior of search functionality. See the findQualifiers appendix for more information.
tModelBag	This is a list of tModel uuid_key values that represents the technical fingerprint of a bindingTemplate structure contained within the businessService specified by the serviceKey value. Only bindingTemplates that contain all of the tModel keys specified will be returned (logical AND). The order of the keys in the tModel bag is not relevant.

Returns

This API call returns a bindingDetail message upon success. In the event that no matches were located for the specified criteria, the bindingDetail structure returned will be empty (i.e., it contains no bindingTemplate data.) This signifies a zero match result. If no arguments are passed, a zero-match result set will be returned.
In the event of an overly large number of matches (as determined by each Operator Site), or if the number of matches exceeds the value of the maxRows attribute, the Operator site will truncate the result set. If this occurs, the response message will contain the truncated attribute with the value "true".

Caveats

If any error occurs in processing this API call, a dispositionReport element will be returned to the caller within a SOAP Fault. The following error number information will be relevant:

E_invalidKeyPassed	This signifies that the uuid_key value passed did not match with any known serviceKey or tModelKey values. The error structure will signify which condition occurred first, and the invalid key will be indicated clearly in text.
E_unsupported	This signifies that one of the findQualifier values passed was invalid. The invalid qualifier will be indicated clearly in text.

G1132

Statement:

Implement the data tier using **commercial off-the-shelf** (**COTS**) **relational database management system** (**RDBMS**) products that implement the **SQL** standard.

Rationale:

COTS RDBMS products are technically mature, and their capabilities are continually expanding (to include capabilities such as row-level locking, stored procedures, triggers, and high-level language interfaces). Moreover, there is a large technical community able to develop and maintain data systems based on these products. It is likely that a COTS RDBMS will provide many of the data tier capabilities a developer requires.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Design Tenet: Enterprise Service Management
Database Implementations
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Interoperability

Evaluation Criteria:

1) Test: [G1132.1]

Is the proposed COTS RDBMS product a readily available and supportable COTS product that implements the SQL standard?

Procedure:

Verify that the COTS RDBMS product is widely in use in the DoD environment (e.g., Oracle, SQL Server, or DB2), has a large support community, and is likely to be supported for the lifecycle of the project.

Example:

None

G1141

Statement:

Use standard **data models** developed by **Communities of Interest** (**COI**) as the basis of program or project data models.

Rationale:

Standard **data models** are under development in many areas of the DoD and will be stored in and made available from DoD **metadata** repositories. The use of these models or portions thereof supports interoperability among applications. The **C2IEDM** data model, used in the **Command and Control** area, is an example of one of these standard data model development efforts.

Referenced By:

Database Development
Design Tenet: Service-Oriented Architecture (SOA)
Reading/Writing Objects within a DDS Domain
Design Tenet: Accommodate Heterogeneity
Interoperability
Data Modeling
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1141.2]

If the system is a command-and-control application, has preference been given to the use of the Command & Control Information Exchange Data Model (**C2IEDM**) rather than locally defined values?

Procedure:

Examine the system **data model** and verify that the **C2IEDM** data model has been incorporated.

Example:

None

2) Test: [G1141.1]

Have standard **data models** been considered for use in the system?

Procedure:

Determine whether standard DoD **data models** exist for the technical areas accommodated in the system requirements. Verify that data model the developed for the application accommodates the use of these data models.

Example:

None

G1144

Statement:

Develop two-level database models: one level captures the **conceptual** or logical aspects, and the other level captures the **physical** aspects.

Rationale:

There are a number of modeling tools available that support entity-relationship diagram (ERD) development. Developers can use these tools to create conceptual/logical models that are independent of the **DBMS** in which the system is implemented and to develop the physical models that are translated directly into data definition language (DDL), the **SQL** code used to create the database. Using a conceptual/logical model permits implementation or reuse of a complex ERD on multiple **DBMS** products.

Referenced By:

Design Tenet: Open Architecture
Reusability
Data Modeling
Database Development
Design Tenet: Service-Oriented Architecture (SOA)
Composeability

Evaluation Criteria:

1) Test: [G1144.1]

Have separate **conceptual**/logical and **physical** models been developed?

Procedure:

Verify the presence of a conceptual/logicalmodel0 and a physical model.

Example:

None

G1146

Statement:

Include information in the **data model** necessary to generate a **data dictionary**.

Rationale:

A **data dictionary** is an integral part of every system including databases. A description of each data item and the units in which the contents are measured are essential. **Data modeling** tools provide a mechanism for storing information necessary to produce a **data dictionary**.

Referenced By:

RDBMS Internals
Reading/Writing Objects within a DDS Domain
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability

Evaluation Criteria:

1) Test: [G1146.1]

Does the data model include description information?

Procedure:

Examine the physical data model.

Example:

None

G1147

Statement:

Use **domain analysis** to define the constraints on input data validation.

Rationale:

Domain analysis is an integral part of any data system including databases. Domains describe the set or range of values that are acceptable for a specific data item. These include, at a minimum the following:

- Data type
- Precision
- Minimum
- Maximum
- Length

These values are used to validate the data.

In the database, the range checking is done via check constraints on the data item. These **check constraints** are generated from the **physical data model** as part of the DDL.

Referenced By:

Database Development
Data Modeling
Design Tenet: Service-Oriented Architecture (SOA)
Reading/Writing Objects within a DDS Domain
Maintainability
Validate Input

Evaluation Criteria:

1) Test: [G1147.1]

Does the data model include include constraints derived from domain analysis?

Procedure:

Examine the physical data model.

Example:

None

G1148

Statement:

Normalize data models.

Rationale:

Normalization is a central **tenet** of **relational database** theory. It is also part of **OOA**.

A database should usually be normalized to at least third normal form. Although there are seven normal forms, normalization beyond third normal form is rarely considered in practical database design.

Objects developed in the absence of data normalization are prone to unnecessary complexity required to keep multiply copies of data.

Referenced By:

Reading/Writing Objects within a DDS Domain
Database Development
Maintainability
Data Modeling
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1148.1]

Is the database design in third normal form?

Procedure:

Examine the conceptual/logical **data model**.

Example:

None

G1151

Statement:

Define declarative **foreign keys** for all relationships between tables to enforce **referential integrity**.

Rationale:

Foreign Key constraints enforce referential integrity. The principle of referential integrity requires that the foreign key values of a child table are either null or match exactly those of the **primary key** in the parent table.

Referenced By:

Database Development
Design Tenet: Service-Oriented Architecture (SOA)
RDBMS Internals
Maintainability

Evaluation Criteria:

1) Test: [G1151.1]

Have foreign-key constraints been incorporated into the database?

Procedure:

Examine the database to determine whether foreign-key constraints have been included in the database creation scripts and created in the database.

Example:

None

G1153

Statement:

Separate application, presentation, and data tiers.

Rationale:

Separation into tiers allows for the separate maintenance of each tier as long as the interface between tiers does not change. It also allows for multiple implementations of a layer to meet
different requirements. This supports technology refresh and certain requirements changes.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Maintainability
Design Tenet: Scalability
RDBMS Internals
Composeability
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1153.1]

Does the program, project or initiative architecture support clear boundaries between application layers, e.g. data, presentation, and business logic layers.

Procedure:

Examination of the program, project or initiative architecture and evaluate the degree to which it supports clear boundaries between applications layers such as data, and presentation layers.

Verify that the system design accommodates a multi-tier architecture.

Example:

The use of web services is one means of separating the presentation layer from business logic and data layers.

G1154

Statement:

Use **stored procedures** for operations that are focused on the insertion and maintenance of data.

Rationale:

Current software design methodologies and architectures call for the implementation of an n-tiered architecture with business rules in the middle tier and data stored in a separate data tier. When multiple applications access a common database, however, the rules may be best located at the data-tier level. Otherwise, changes in one application would have to be coordinated across all applications.

Referenced By:

RDBMS Internals
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability
Design Tenet: Make Data Trustable

Evaluation Criteria:

1) Test: [G1154.1]

Are database triggers used?

Procedure:

Check for stored procedures that are triggered on insertion, deletion, and update events.

Example:

```
CREATE TRIGGER PersonCheckAge
AFTER INSERT OR UPDATE OF age
ON Person
FOR EACH ROW
BEGIN
  IF (:new.age < 0) THEN
    RAISE_APPLICATION_ERROR
      ( -20000,
        'no negative age allowed'
      );
  END IF;
END;.
```

G1155

Statement:

Use **triggers** to enforce **referential** or **data integrity**, not to perform complex **business logic**.

Rationale:

Triggers are fired on events. Current software design methodologies and architectures call for the implementation of an n-tiered architecture with business rules in the middle tier and data stored in a separate data tier. Implementing business logic in triggers, as well as in the middle tier, violates this concept.

Referenced By:

Composeability
Design Tenet: Make Data Trustable
Design Tenet: Service-Oriented Architecture (SOA)
RDBMS Internals
Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test: [G1155.1]

Has business logic been incorporated into database triggers?

Procedure:

Examine the database trigger code to determine whether business logic or calls to stored procedures incorporating business logic have been coded into them.

Example:

None

G1190

Statement:

Use a build tool.

Rationale:

A build tool allows for the encapsulation of building instructions into machine-readable files or sets of files. The instructions can be successfully and consistently repeated.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Automate the Software Build Process

Evaluation Criteria:

1) Test: [G1190.1]

Does the program or project use a build tool?

Procedure:

Identify which build tool the program or project is using.

Example:

None.

G1202

Statement:

Use the **CORBA Portable Object Adapter** (**POA**) instead of the **Basic Object Adapter** (**BOA**).

Rationale:

The CORBA Basic Object Adapter (BOA) was the CORBA Version 1 specification for the client-server object capability. The BOA specification was found to be so incomplete that vendor-specific interpretations were required for operable implementation. In CORBA Version 2, the Portable Object Adapter (POA) was significantly more complete and flexible. In the current marketplace, POA implementations are standard and, in quality implementations, are not vendor-specific. Consequently, using POA eliminates one significant area of vendor-specific coding.

BOA	***POA***
• Focuses on CORBA server implementations and not CORBA object implementations • Naming convention issues on server side • Tightly coupled to **ORB** implementation • Non-standardized way to connect to ORB • Four activation models for server processes	• Services for lifecycle management • Abstract layer between ORB and object • Standard, portable interface for communicating with ORB runtime • Two servant incarnation styles

Referenced By:

Interoperability
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Composeability
CORBA

Evaluation Criteria:

1) Test: [G1202.1]

Does any CORBA application code reference the **`CORBA::BOA`** identifier.

Procedure:

Review the code for the use of the **`CORBA::BOA`** identifier.

Example:

BOA Coding Example

Client Side

The code below shows a C++ CORBA client BOA initialization for the ORBIX ORB. Other ORB vendors may have different initialization sequences.

```
int main
  ( int argc,
    char **argv
  )
{ MyServer_var MyVar;
  CORBA::ORB_ptr myOrbPtr
    = CORBA::ORB_init(argc, argv,"Orbix");
  try
  { // The default is the local host:
    MyVar = MyServer::_bind(":ServerName");
  } // End try
  catch ( CORBA::SystemException &sysEx )
  { cerr << "Unexpected system exception" << endl;
    cerr <<&sysEx;
    exit(1);
  } // End CORBA::SystemException
  catch(...)
  { // an error occurred while trying
    // to bind to the grid object.
    cerr << "Bind to object failed" << endl;
    cerr << "Unexpected exception " << endl;
    exit(1);
  } // End catch ...
} // End main
```

Server Side

Use the code below as a model. This example shows a C++ CORBA server BOA init for the ORBIX ORB. For BOA, other ORBS will have a different initialization sequence.

```
try
{ MyObject::myOrb_
    = CORBA::ORB_init(argc, argv, "Orbix");
  MyObject::myboa_
    = MyObject::myOrb_->BOA_init(argc, argv, "Orbix_BOA");
} // End try
catch ( CORBA::SystemException &sysEx )
{ //some exception handling code
} // End catch
try
{ NoeLoggerCfg::myboa_->impl_is_ready("MyServiceName",
  CORBA::ORB::INFINITE_TIMEOUT);
} // End try
catch ( CORBA::SystemException &sysEx )
{ //exception handling code
}
```

POA Coding Example

Client Side

This example shows a C++ CORBA client POA init for the ORBIX ORB. For BOA, other ORBS will have a different initialization sequence.

```
int main
  ( int argc,
    char **argv
  )
{ CORBA::ORB_var myOrb = CORBA::ORB_init(argc, argv);
  try
  { CORBA::Object_var obj
      = ... // however you get the object reference
    if(CORBA::is_nil (obj))
    { cerr << "Nil object reference" << endl;
      throw 0;
    } // End if
  } // End try
  catch ( CORBA::SystemException &sysEx )
  { cerr << "Unexpected system exception" << endl;
    cerr <<&sysEx;
    exit(1);
  } // End catch CORBA::SystemException
  catch ( ... )
  { cerr << "Unexpected system exception" << endl;
    exit(1);
  } // End catch ...
  myinterface::myobject_var myvar;
  try
  { myvar = myinterface::myobject::_narrow(obj);
  } // End try
  catch ( CORBA::SystemException &sysEx)
  { cerr << "Unexpected system exception" << endl;
    cerr <<&sysEx;
    exit(1);
  } // End catch CORBA::SystemException
} // End main
```

Server Side

Use the code below as a model. This example shows a C++ CORBA server POA init for the ORBIX ORB. For POA, other ORBS will have a different initialization sequence.

```
int main
  ( int argc,
    char *argv[ ]
  )
{ try
  { // initialize the ORB
    orb_var orb  = CORBA::ORB_init(argc, argv, "Orbix");
    // obtain an object reference for the root POA
    object_var obj
      = orb->resolve_initial_references ("RootPOA");
    POA_var poa = POA::_narrow(obj);
    // incarnate a servant
    My_Servant_Impl servant;
    // Implicitly register the servant with the root POA
    obj =  servant._this ();
    //start the POA listening for requests
    poa -> the_POAManager ()->activate ();
    //run the orb's event loop
    orb->run ();
  } // End try
  catch ( CORBA::SystemException &sysEx )
  { // some exception handling code
  } // End catch
} // End main
```

G1203

Statement:

Localize frequently used **CORBA**-specific code in **modules** that multiple applications can use.

Rationale:

In a family of applications, similar patterns of CORBA Object Request Broker (**ORB**) invocation sequences frequently arise. This is common in service object initialization, policy association, discovery, binding, and release handling. Implementing this functionality in a utility library paradigm localizes the code to reduce maintenance and facilitate extensibility, and assures consistency across the family of applications.

Referenced By:

Maintainability
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)
Reusability
Extensibility
CORBA
Design Tenet: Open Architecture
Interoperability

Evaluation Criteria:

1) Test: [G1203.2]

Do the standard object policy association CORBA invocations occur in more than one module?

Procedure:

The presence of "**`CORBA::PolicyList`**" in C++ indicates policy presence.

Example:

None

2) Test: [G1203.1]

Do the standard object initialization CORBA invocations occur in more than one module?

Procedure:

The presence of "**`CORBA::ORB_var`**" or "**`CORBA::ORB_init`**" in C++ indicates ORB initialization. The presence of "**`CORBA::Object_var`**" in C++ indicates ORB access.

Example:

None

3) Test: [G1203.3]

Do the standard object policy association CORBA invocations occur in more than one module?

Procedure:

The presence of "**`CORBA::PolicyList`**" in C++ indicates policy presence.

Example:

None

4) Test: [G1203.4]

Do the standard object discovery CORBA invocations occur in more than one module?

Procedure:

The presence of "**`Resolve_NamingService()`**"in C++ indicates intended access to one of CORBA's discovery capabilities.

Example:

None

5) Test: [G1203.5]

Do the standard object binding and release CORBA invocations occur in more than one module?

Procedure:

The presence of "**`::_narrow(obj.in())`**" or "**`CORBA::is_nil(`**" in C++ indicates activity associated with obtaining and validating an object binding to a legitimate reference. The presence of "**`CORBA(release)(`**" in C++ indicates intended release of a CORBA-bound object reference.

Example:

None

G1204

Statement:

Create configuration services to provide distributed user control of the appropriate configuration parameters.

Rationale:

For user-modifiable configuration settings that are intended to be accessible by distributed processes at runtime, the appropriate mechanism for implementation involves **CORBA** services. The first form is a network service to be invoked as a client by the target system application at initialization. This can support a consistent, network-wide distribution of startup parameters. The second form is a service implemented by the target application which allows communication to the application during execution (after startup). This allows **real-time** configuration changes for matters such as Portable Object Adapter (**POA**) instantiation threading policies to address load management.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Design Tenet: Open Architecture
Maintainability
Design Tenet: Decentralized Operations and Management
CORBA

Evaluation Criteria:

1) Test: [G1204.1]

Is a service defined in the IDL to obtain the configuration parameters?

Procedure:

Review the code for a service that can be used to obtain configuration.

Example:

The following code is an example of a CORBA server that instantiates a configuration service. The service manages the individual configuration parameters for the servers on the ORB.

Ada Example

```
CORBA.ORB.IIOP_English;
pragma Elaborate_All(CORBA.ORB.IIOP_English);
with CORBA ;
with CORBA.BOA ;
with CORBA.ORB ;
with CORBA.Object ;
with Configuration.Impl ;
with Configuration.Helper ;
with Ada.Exceptions ;
with Ada.Text_IO ;
with my_CORBA ;
with Event_Ada_API ;
procedure Configuration_Server is
    -- required for OrbExpress
    First_Variable : CORBA.ORB.Life_Span ;
    -- declare the object instance
    Configuration_Object : Configuration.Ref ;
    --variables needed for ior writing
    No_Timeout : constant := 0.0;
    Config_Name : constant String
      := Configuration.Helper.Simple_Name ;
    Config_Host : Corba.String ;
    Config_Port : Corba.String ;
begin -- Configuration_Server
  -- create (and initialize) the object
  -- config file is read and the port needed
  -- is in there
  Configuration_Object
    := Configuration.Impl.Create(Config_Name) ;
  GET_HOSTNAME:
  begin
    Config_Host
      := Configuration.Get_String
         ( Self => Configuration_Object,
           Name => Corba.To_Corba_String
                     ( "Local_Host_Shortname" )
         );
  exception -- GET_HOSTNAME
    when others =>
      Ada.Text_IO.Put_Line
        ( "ERROR: Missing parameter"
         & "<Local_Host_Shortname> "
         & "in the config_parameters.txt file."
        );
  end GET_HOSTNAME;
  GET_CS_PORT:
  begin
    Config_Port
      := Configuration.Get_String
         ( Self => Configuration_Object,
           Name => Corba.To_Corba_String
                     ( "Config_Service_Port" )
         );
  Exception -- GET_CS_PORT
    when others =>
      Ada.Text_IO.Put_Line
        ( "ERROR: Missing parameter "
         & "<Config_Service_Port> "
         & "in the config_parameters.txt file."
        );
  end GET_CS_PORT;
  Ada.Text_IO.Put_Line
    ( "Host => "
        & Corba.To_Standard_String(Config_Host)
        & " Port => "
        & Corba.To_Standard_String(Config_Port)
    );
  --timeout 0 so we can write IOR out
  CORBA.BOA.Impl_Is_Ready
      ( Time_Out             => No_Timeout,
        Server_Instance_Name => Config_Name,
        Listen_On_Endpoints  =>
          "tcp://"
          & Corba.To_Standard_String(Config_Host)
          & ":"
          & Corba.To_Standard_String(Config_Port)
       );
      -- ---------------------------------------------
      -- HERE IS WHERE CODE FOR THE IOR TO BE
      -- USED ON THE C++ ORB
      -- ---------------------------------------------
  -- get the IOR and write it to disk
  my_CORBA.Write_IOR_To_File
    ( Server_Name => Config_Name,
      Server_Ref  =>
        CORBA.Object.Ref(Configuration_Object)
    );
  READY_BLOCK:
  begin
    -- notify subscribers of availability
    -- of configuration parameters via the
    -- event service
    Event_Ada_API.Send
      ( Channel_Name => "Config_Channel",
        Event        => "Configuration Service Ready."
      );
  Exception - READY_BLOCK
    when others =>
      Ada.Text_IO.Put_line
        ( "Configuration_Server : "
         & Exception sending ready signal."
        );
  end READY_BLOCK;
  Ada.Text_IO.Put_line
    ( "Configuration_Server : "
     & Configuration Service Ready."
    );
  CORBA.BOA.Impl_Is_Ready
    ( Time_Out             => CORBA.Infinite_Timeout,
      Server_Instance_Name => Config_Name
    ) ;
exception -- Configuration_Server
  when X_Other: others =>
    Ada.Text_IO.Put_line
      ( "Configuration_Server : "
       & Ada.Exceptions.Exception_Name(X_Other)
      );
end Configuration_Server ;
```

C++ Example

The following code snippets depict a C++ server that instantiates a version collection service for an About box. It uses the IORs from the servers on the Ada ORB via the IOR files, and invokes those objects to get version information. It uses the utility templates for binding. It exemplifies the approach described in Encapsulate CORBA ORB operations for C++.

Note: *This was done on the ORBIX C++ and Ada ORBs.*

```
#include <iostream.h>
#include <rw/cstring.h>
#ifndef _STDIO_H
#include <stdio.h>
#endif
#ifndef _STRING_H
#include <string.h>
#endif
#ifndef _STDLIB_H
#include <stdlib.h>
#endif
#ifndef _ASSERT_H
#include <assert.h>
#endif
// Include files for all the objects desired for
// collecting version information
//Ada configuration service
#ifndef configuration_hh
#include <configuration.hh>
#endif
// include files for other desired services;
// removed for brevity
// other support objects and utilities
#ifndef _CORBA_UTILS__
#include <corba_utils.h>
#endif
#ifndef __LOG_API_H__
#include <log_api.h>
#endif
#ifndef _VERSION_AGENT_GLOBALS_H_
```

```
#include "version_agent_globals.h"
#endif
const RWCString  Version_Agent_i::MSG_VERSION_NOT_FOUND_
  = "Version Info. not found for ";
const CORBA::ULong Version_Agent_i::MAXSERVERS_
  = 12;
Version_Agent_i:: Version_Agent_i(): theVersionInfoPtr_(0)
{ theVersionInfoPtr_
     = new versionInfoType(MAXSERVERS_);
  theVersionInfoPtr_->length(MAXSERVERS_);
} // End constructor
Version_Agent_i:: ~Version_Agent_i()
{ // Do nothing
} // End destructor
/***********************************************************
FUNCTION NAME: createVersions
PURPOSE: helper function that gets the version info
INPUT:
OUTPUT:
***********************************************************/
void Version_Agent_i::createVersions ()
{ char *iorString;
  int bBindOk = 0;
  int versionCnt = 0;
  versionInfoType* rl = theVersionInfoPtr_;
  CORBA::ULong MAXSERVERS Version_Agent_i::MAXSERVERS_;
  // server variables for all the objects desired
  // for collecting version information
  // most declarations removed for brevity
    EventServiceFactory_var es_var;
  // Ada configuration service
    Configuration_var cfg_var;
  // === load the versions of the individual components
  // Code for other services removed for brevity
  // This is an ADA service using the IOR string
  {  //****************** config service ***************
    logMsg
      ( "get config service version",
        Log_Api::DEBUG_1_MSG
      );
    RWCString errMsg
      ( Version_Agent_i::MSG_VERSION_NOT_FOUND_.data()
      );
    errMsg.append ( "Configuration Service" );
    // here we get the IOR from the ADA orb using
    // the helper methods
      iorString = getIorFile("Configuration");
    //template class to hide binding issues to the ADA ORB
    If ( iorString )
    { Ada_Binder < Configuration,
      Configuration_var > bo ( iorString );
      bBindOk = bo.bindToAda(&cfg_var) ;
      // get the version info and load it
      If ( bBindOk
          && !( CORBA::is_nil(cfg_var))
         )
     { try
       { char* str = cfg_var->version();
         if ( str )
         { (*theVersionInfoPtr_)[versionCnt]
             = CORBA::string_dup(str);
          delete str;
         } // End if
         else
         { (*theVersionInfoPtr_)[versionCnt]
             = CORBA::string_dup(errMsg.data());
         } // End else
       } // End try
       catch(...)
       { (*theVersionInfoPtr_)[versionCnt]
           = CORBA::string_dup(errMsg.data());
       } // End catch
       cfg_var->_closeChannel();
     } // End if
     else
```

```
      { (*theVersionInfoPtr_)[versionCnt]
           = CORBA::string_dup(errMsg.data());
      } // End else
      if(iorString)
      { free (iorString);
        iorString = NULL;
      } // End if
    } //endif iorstring
    else
    { (*theVersionInfoPtr_)[versionCnt]
        = CORBA::string_dup(errMsg.data());
    } // End else
    //leaving scope releases the corba object
  } //end cfg_svf
  bBindOk = 0;
  versionCnt++;
  assert(versionCnt <= MAXSERVERS);
} // End createVersions
/***********************************************************
FUNCTION NAME: start
PURPOSE:  handle startup specific stuff
INPUT:
OUTPUT:
***********************************************************/
void Version_Agent_i:: start
  ( CORBA::Environment &IT_env
  ) throw (CORBA::SystemException)
{ //get all the version info
  createVersions();
} // End start
/***********************************************************
FUNCTION NAME: stop
PURPOSE:  handle stop specific stuff
INPUT:
OUTPUT:
***********************************************************/
void Version_Agent_i:: stop
  ( CORBA::Environment &IT_env
  ) throw (CORBA::SystemException)
{ // Release info
  // Let CORBA time out the service
  logMsg ( "stop received" );
  VersionAgentGlobals::myboa->setNoHangup ( 0 );
  VersionAgentGlobals::myboa->deactivate_impl
    ( "Version_Agent" );
} //end version impl
```

G1205

Statement:

Use non-source code persistence to store all user-modifiable **CORBA** service configuration parameters.

Rationale:

For user-modifiable configuration settings that are host-specific and that are not intended to be accessible by distributed processes at runtime, the appropriate mechanism for implementation involves local persistent storage. The appropriate form of local storage depends on the local host architecture and may be file- or host-DBMS oriented. It is important that such parameters are not stored in source code that requires build processes for modification.

For **SOA** services, configuration parameters relating to invoked services should not be service-host-specific at the invoking client application.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Maintainability
CORBA

Evaluation Criteria:

1) Test: [G1205.1]

Are there any user-modifiable configuration parameters hard coded in the non-auto-generated files?

Procedure:

Inspect the code for constant strings or constants that contain configuration parameters.

Example:

None

G1208

Statement:

Add new functionality rather than redefining existing interfaces in a manner that brings incompatibility.

Rationale:

By not replacing old methods of objects, library functionality consumers can continue to operate and not be forced to upgrade.

Referenced By:

Public Interface Design
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1208.1]

Are methods that are being replaced marked with deprecated tags?

Procedure:

Check revision history to make sure that methods are deprecated and not removed unless they have expired. "Expired" means that they have passed the expected shelf life, as defined by the project standards or other standards documentation.

Example:

None

2) Test: [G1208.2]

Do new methods being added contain information on methods they are replacing?

Procedure:

Check to make sure newly added methods contain information and rationale on the methods they are replacing.

Example:

None

G1209

Statement:

For Java, use **JDK** logging facilities.

Rationale:

Java has a built-in logging framework that is portable across platforms, projects, and installations.

Referenced By:

Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Java EE Environment
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test: [G1209.1]

Does the application use anything other than the specified logging frameworks?

Procedure:

Check for use of logging frameworks other than the JDK.

Example:

None

G1210

Statement:

For **.NET**, use Debug and Trace from the **`System.Diagnostics`** **namespace**.

Rationale:

.NET has a built-in logging framework that is portable across .NET projects and installations.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Interoperability
Design Tenet: Enterprise Service Management
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
.NET Framework

Evaluation Criteria:

1) Test: [G1210.1]

Does the application use anything other than the specified logging frameworks?

Procedure:

Check for use of logging frameworks other than **`System.Diagnostics`**.

Example:

None

G1213

Statement:

Provide an architecture design document.

Rationale:

An architectural design document provides evaluators with a roadmap of the application. This helps evaluators verify that the application follows guidance such as using the Model View Controller model.

Referenced By:

Design Tenet: Open Architecture
Public Interface Design
Maintainability

Evaluation Criteria:

1) Test: [G1213.1]

Do the project deliverables for evaluation include a document that contains the architectural design of the application?

Procedure:

See if an architectural design document exists.

Example:

None

G1214

Statement:

Provide a document with a plan for **deprecating** obsolete **interfaces**.

Rationale:

This information allows users to phase out deprecated interfaces. For instance, Sun plans to maintain backward compatibility for the **JDK** for seven years. This means developers can count on deprecated methods not being removed for seven years.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Public Interface Design

Evaluation Criteria:

1) Test: [G1214.1]

Do the project deliverables for evaluation include a document that contains a plan for deprecating obsolete interfaces?

Procedure:

See if a document with a plan for deprecating obsolete interfaces exists.

Example:

None.

G1215

Statement:

Provide a coding standards document.

Rationale:

The standards ensure a consistent code base. A coding standards document defines rules to keep code readable, maintainable, and secure.

Referenced By:

Public Interface Design
Design Tenet: Open Architecture
Apply Secure Coding Standards
Maintainability

Evaluation Criteria:

1) Test: [G1215.1]

Do the project deliverables for evaluation include a coding standards document?

Procedure:

See if a coding standards document exists.

Example:

None

G1216

Statement:

Provide a software release plan document.

Rationale:

The release plan document ensures that there is a formal process for releasing the software. It includes a description of how to acquire the software from the software configuration management (SCM) repository and how to build, label, and release it.

Referenced By:

Public Interface Design
Maintainability
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1216.1]

Do the project deliverables for evaluation contain a release plan document?

Procedure:

See if a software release plan exists.

Example:

None

G1217

Statement:

Develop and use externally configurable components.

Rationale:

To be portable and to accommodate reuse, components must be configurable using external descriptors usually defined in **XML**. Examples of things that might need to be configured include the following:

- A data source for the component to obtain a Java Database Connection (**JDBC**)
- The location of a service with which the component must communicate
- The location of implementation classes that the component uses

Referenced By:

Implement a Component-Based Architecture
Design Tenet: Accommodate Heterogeneity
Design Tenet: Service-Oriented Architecture (SOA)
Reusability
Design Tenet: Open Architecture
Maintainability

Evaluation Criteria:

1) Test: [G1217.1]

Are deployment descriptors used?

Procedure:

Check for the existence of deployment descriptors in the appropriate directories. Usually the file is named **`web.xml`**.

Example:

None

G1218

Statement:

Use a build tool that supports operation in an automated mode.

Rationale:

During testing, human interaction can be a cause of error and unrepeatable results. Operating in automated mode can eliminate these errors.

Referenced By:

Automate the Software Build Process
Design Tenet: Open Architecture
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1218.1]

Does the tool have a build all target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to build the entire project, system, or application.

Example:

None

G1219

Statement:

Use a build tool that checks out files from configuration control.

Rationale:

To make sure all the parts of the build are under configuration control, compare all files with the configuration baseline, and download the appropriate files.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Maintainability
Automate the Software Build Process

Evaluation Criteria:

1) Test: [G1219.1]

Does the tool have a checkout target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to check out the entire project, system, or application.

Example:

None

G1220

Statement:

Use a build tool that **compiles** source code and dependencies that have been modified.

Rationale:

To limit the changes made between builds, only compile code that has been modified. If there are no intermediate files, then compile all files.

Referenced By:

Automate the Software Build Process
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Maintainability

Evaluation Criteria:

1) Test: [G1220.1]

Does the tool have a compile target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to compile the entire project, system, or application.

Example:

None

2) Test: [G1220.2]

Do all the intermediate files (e.g., **`.obj`** or **`.class`**) have the same date and time stamps?

Procedure:

Scan the files for date and time stamps.

Example:

None

G1221

Statement:

Use a build tool that creates libraries or archives after all required compilations are completed.

Rationale:

Libraries should be able to be recreated independently of any executables and should always verify that any intermediate files are not stale.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Automate the Software Build Process
Maintainability

Evaluation Criteria:

1) Test: [G1221.1]

Does the tool have a generate library target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to generate the composing libraries or archives.

Example:

None

G1222

Statement:

Use a build tool that creates executables.

Rationale:

An executable is dependent on many files, including source files, intermediate files, and libraries or archives. The building of the executable must support a control process that includes configuration management, compiling, and testing.

Referenced By:

Automate the Software Build Process
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Maintainability

Evaluation Criteria:

1) Test: [G1222.1]

Does the tool have an executable target?

Procedure:

Check the build scripts or build tool descriptors for the ability to build the executables for the entire project, system, or application.

Example:

None

G1223

Statement:

Use a build tool that is capable of running unit tests.

Rationale:

All code should be able to be tested independently of creating intermediate files, libraries, or executables.

Tests should be unit tests as well as system-level tests.

Referenced By:

Automate the Software Build Process
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1223.1]

Does the tool have a test target?

Procedure:

Check the build scripts or descriptors of the build tool for the ability to test the entire project, system, or application.

Example:

None

G1224

Statement:

Use a build tool that cleans out intermediate files that can be regenerated.

Rationale:

For security reasons, all files that comprise the build need to be under configuration control. Cleaning out all files is essential in ensuring that only approved code is incorporated into the build.

Referenced By:

Automate the Software Build Process
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Maintainability

Evaluation Criteria:

1) Test: [G1224.1]

Does the tool have a clean target?

Procedure:

Check the build scripts or descriptors for the build tool for the ability to remove the entire project, system, or application files.

Example:

None

G1225

Statement:

Use a build tool that is independent of the **Integrated Development Environment**.

Rationale:

Some build tools are tightly coupled with an **Integrated Development Environment** (**IDE**) that causes vendor lock-in and license issues when the software is delivered to the Government.

Referenced By:

Maintainability
Automate the Software Build Process
Design Tenet: Open Architecture
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1225.2]

Is the build tool one of the recognized standards, such as ant?

Procedure:

Check for files named **`build.xml`**.

Example:

None

2) Test: [G1225.3]

Is the build tool one of the recognized standards, such as **`make`** or **`nmake`**?

Procedure:

Check for files with the name **`makefile`**.

Example:

None

3) Test: [G1225.1]

Does the build tool require a license?

Procedure:

Check for files with the name **`makefile`**.

Example:

None

G1236

Statement:

Do not **hard-code** the **endpoint** of a **Web service** vendor.

Rationale:

An endpoint is the URL or location of the **Web service** on the **Internet**. A major benefit of Web services is the ability to relocate a Web service to another location or dynamically discover and use a Web service using registry facilities. Some Web service vendors hard-code the URL of the Web service which causes maintenance and portability problems.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Insulation and Structure
Maintainability

Evaluation Criteria:

1) Test: [G1236.1]

Are there any hard-coded Web service vendor endpoints in the client code?

Procedure:

Parse the code and look for hard-coded endpoints. These endpoints look just like a normal HTTP Web address.

Example:

None

G1237

Statement:

Do not **hard-code** the configuration data of a **Web service** vendor.

Rationale:

Some vendors generate code that passes Web service vendor-specific configuration data during initialization or startup. This reduces the portability of the code and can cause maintenance problems later.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity
Interoperability
Insulation and Structure
Maintainability

Evaluation Criteria:

1) Test: [G1237.1]

Is there any Web service vendor-specific configuration data in the client code?

Procedure:

Parse the code and look for hard-coded configuration data that might be used to configure the vendor's Web service.

Example:

None

G1239

Statement:

Use design patterns (e.g., **facade**, **proxy**, or **adapter**) or property files to isolate vendor-specifics of vendor-dependent connections to the enterprise.

Rationale:

This isolation increases maintainability. Guidance G1071 asserts that vendor-neutral connection mechanisms should be used. When vendor-specific connection mechanisms are unavoidable, this guidance will apply.

Referenced By:

Design Tenet: Open Architecture
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
JNDI Security

Evaluation Criteria:

1) Test: [G1239.1]

Is the connection mechanism vendor-dependent?

Procedure:

Examine the source code for vendor-specific imports or includes.

Make sure that all references to the vendor-specific connection mechanisms are isolated to a single class (like a helper) or set of methods that are used as part of an isolation design pattern such as facade, proxy, or adapter.

Also, look for hard-coded vendor-specific connection strings.

Example:

None

G1245

Statement:

Isolate the **Web service portlet** from platform dependencies using the **Web Services for Remote Portlets** (**WSRP**) Specification protocol.

Rationale:

The **OASISWSRP** 1.0 Specification accounts for the fact that **producers** and **consumers** may be implemented on very different platforms, such as a Java EE-based Web service, a Web service implemented on the Microsoft .Net platform, or a **portlet** published directly by a **portal**.

Referenced By:

Web Portals
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Design Tenet: Open Architecture
Design Tenet: Decentralized Operations and Management

Evaluation Criteria:

1) Test: [G1245.3]

Does the Web service implement the WSRP Portlet Configuration interface?

Procedure:

Look for the occurrence of the **`getService`**, **`getPortletDescription`**, **`clonePortlet`**, **`destroyPortlets`**, **`setPortletProperties`**, **`getPortletProperties`** and **`getPortletPropertyDescription`** methods as defined in the OASIS WSRP Portlet Configuration API Specification.

Example:

```
public static PortletManagementService getService
  ( java.lang.String baseEndpoint
  ) throws java.lang.Exception
public PortletDescriptionResponse getPortletDescription
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] desiredLocales
  ) throws java.lang.Exception
public PortletContext clonePortlet
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext
  ) throws java.lang.Exception
public DestroyPortletsResponse destroyPortlets
  ( RegistrationContext registrationContext,
    java.lang.String[] portletHandles
  ) throws java.lang.Exception
public PortletContext setPortletProperties
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    PropertyList propertyList
  ) throws java.lang.Exception
public PropertyList getPortletProperties
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] names
  ) throws java.lang.Exception
public PortletPropertyDescriptionResponse getPortletPropertyDescription
  ( RegistrationContext registrationContext,
    PortletContext portletContext,
    UserContext userContext,
    java.lang.String[] desiredLocales
  ) throws java.lang.ExceptionThrows
```

2) Test: [G1245.1]

Does the Web service implement the WSRP Markup interface?

Procedure:

Look for the definition of the **`getMarkup`**, **`performBlockingInteraction`**, **`initCookie`** and **`releaseSessions`** methods as defined in the OASIS WSRP Markup API Specification.

Example:

```
public MarkupResponse getMarkup
   ( RegistrationContext registrationContext,
     PortletContext portletContext,
     RuntimeContext runtimeContext,
     UserContext userContext,
     MarkupParams markupParams
   ) throws java.lang.Exception
public void performBlockingInteraction
   ( RegistrationContext registrationContext,
     PortletContext portletContext,
     RuntimeContext runtimeContext,
     UserContext userContext,
     MarkupParams markupParams,
     InteractionParams interactionParams
   ) throws java.lang.Exception
public Extension[] initCookie
   ( RegistrationContext registrationContext
   )  throws java.lang.Exception
public Extension[] releaseSessions
   ( RegistrationContext registrationContext,
     java.lang.String[] sessionIDs
   ) throws java.lang.Exception
```

3) Test: [G1245.4]

Does the Web service implement the WSRP Registration interface?

Procedure:

Look for the occurrence of the **`getService`**, **`register`**, **`deregister`**, and **`modifyRegistration`** methods as defined in the OASIS WSRP Specification.

Example:

```
public static RegistrationService getService
   ( java.lang.String baseEndpoint
   ) throws java.lang.Exception
public RegistrationContext register
  ( java.lang.String consumerName,
    java.lang.String consumerAgent,
    boolean methodGetSupported,
    java.lang.String[] consumerModes,
    java.lang.String[] consumerWindowStates,
    java.lang.String[] consumerUserScopes,
    java.lang.String[] customUserProfileData,
    Property[] registrationProperties
  ) throws java.lang.Exception
public ReturnAny deregister
  ( java.lang.String registrationHandle,
    byte[] registrationState
  ) throws java.lang.Exception
public RegistrationState modifyRegistration
  ( RegistrationContext registrationContext,
    RegistrationData registrationData
  ) throws java.lang.Exception
```

4) Test: [G1245.2]

Does the Web service implement the WSRP Service Description interface?

Procedure:

Look for the occurrence of the **`getService`**, **`register`**, and **`getServiceDescription`** methods as defined in the OASIS WSRP Service Description API Specification.

Example:

```
public static ServiceDescriptionService getService
  ( java.lang.String baseEndpoint
  ) throws java.lang.ExceptionThrows:
jpublic ServiceDescription getServiceDescription
  ( RegistrationContext registrationContext,
    java.lang.String[] desiredLocales
  ) throws java.lang.Exception
```

G1267

Statement:

Use industry standard HTML data entry fields on Web pages.

Rationale:

Macromedia Flash and Java Applets can also be used for data input but are not HTML standards and tend to decrease the maintainability of a Web site.

Referenced By:

Human Factor Considerations for Web-Based User Interfaces
Design Tenet: Open Architecture
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1267.1]

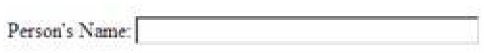
Do any Web pages have data entry fields?

Procedure:

Search all Web pages for the "applet" and "embed" tags. Load each page found in the search by loading and visually inspecting to see if Flash or Applets are used for data entry.

Example:

Correct Usage:


Person's Name:

I1119

Incorrect usage:

Applet	
Flash	

G1268

Statement:

Label all data entry fields.

Rationale:

A label provides the user with a brief description of the text to be entered. Labels are essential for a user to understand the data entry field.

Referenced By:

Human-Computer Interaction
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability

Evaluation Criteria:

1) Test: [G1268.1]

Are all data entry fields labeled?

Procedure:

Search all Web pages for the word "form" and load each resulting Web page in a browser. Visually inspect each data entry field to make sure it has labels.

Example:

None

G1270

Statement:

Include scroll bars for text entry areas if the data buffer is greater than the viewable area.

Rationale:

Scroll bars provide a visual cue to the user that the text extends beyond the viewable area. Scroll bars will appear by default for an HTML text area.

Referenced By:

Interoperability
Human-Computer Interaction
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1270.1]

Do any Web pages turn off scroll bars for text areas?

Procedure:

Search all Web pages and style sheets for the phrase "overflow:hidden" or a form thereof. This turns off scroll bars using styles, but only works in certain browsers. Make sure it is not used.

Example:

Correct Usage

Scroll bars should not be hidden.

Incorrect Usage

Inline style:

```
<html>
<body>
<form>
<textarea style="overflow:hidden"></textarea>
</form>
</body>
</html>
```

External style:

```
textarea.scroll {
   overflow:hidden;
}
```

G1271

Statement:

Provide instructions and **HTML** examples for all style sheets.

Rationale:

An instruction manual will enable developers to use the style sheet correctly and efficiently.

Referenced By:

Browser-Based Clients
Style Sheets
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Reusability
Extensibility
Maintainability
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1271.1]

Are instructions included for each style sheet provided?

Procedure:

Verify that a document is provided that contains instructions and example code for each style provided.

Example:

Correct usage:

```
Cascading style sheet:
.td-items {
   text-align:right;
}
```

Example of usage:

Incorrect usage:
No HTML example explaining style usage.

G1276

Statement:

Do not modify the contents of the Web browser's status bar.

Rationale:

Using the browser's status bar to display text unrelated to status affects interoperability because a user expects the status bar to provide status and nothing else.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Enterprise Service Management
Interoperability
Design Tenet: Accommodate Heterogeneity
Human Factor Considerations for Web-Based User Interfaces
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1276.1]

Do any of the Web pages modify the browser status bar?

Procedure:

Search every Web page for the word "status" and visually inspect each of the search results to see if the status bar has been modified.

Example:

Correct usage:

```
Web pages contain no references to window.status
```

Incorrect usage:

```
window.status = 'text to display in status bar'
```

G1277

Statement:

Do not use tickers on a Web site.

Rationale:

Tickers can irritate the user and use unnecessary bandwidth.

Referenced By:

Human Factor Considerations for Web-Based User Interfaces
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1277.1]

Do any Web pages contain scrolling text?

Procedure:

Most tickers are written using Applets or Flash. Search all Web pages for the "applet" and "embed" tags. Load each page found in the search and visually inspect to make sure no tickers exist.

Example:

Correct usage:

```
No applet or flash references contain tickers.
```

Incorrect usage:

Applet:
```
  applet code="myticker.class" width="200" height="200"
```
Flash:
```
  embed src="myticker.swf" width="200" height="200"
```

G1278

Statement:

Use the browser default setting for links.

Rationale:

Browsers underline links by default. Do not rely on "mouse over" to identify links. Using mouse over to designate links can confuse and slow down infrequent users because they are uncertain which links perform which functions.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Human Factor Considerations for Web-Based User Interfaces
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1278.1]

Do any Web pages or style sheets modify the browser default settings for links?

Procedure:

Search all the Web pages and style sheets for "A:link," "A:visited" and "A:active." Inspect all search results and make sure none of them modify the "A:" items.

Example:

Correct usage:

```
Web pages and style sheets should have no reference to A:link, A:visited or A:active.
```

Incorrect usage:

```
A:link, A:visited, A:active {
   text-decoration:none;
}
```

G1283

Statement:

Use **linked style sheets** rather than embedded styles.

Rationale:

Only by referencing an external file will you be able to update the look of an entire Web site with a single change. Also, by pulling style definitions out of the pages, they (Web pages) will be smaller and faster to download.

Referenced By:

Style Sheets
Maintainability
Reusability
Browser-Based Clients
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Scalability

Evaluation Criteria:

1) Test: [G1283.1]

Does a Web page use the LINK tag to include external style sheets instead of embedding styles?

Procedure:

View the source of the HTML page. The header tag (head) should contain links to external style sheet (.css) files. The header tag should not contain any style tags.

Example:

Correct usage:

External style:

```
<head>
  <link rel=stylesheet href="style.css" type="text/css" media=screen>
  <link rel=stylesheet href="basic.css" type="text/css" media=screen>
</head>
```

Incorrect usage:

Embedded style:

```
<head>
  <style type="text/css">
      td {
      background:#ff0;
      }
  </style>
</head>
```

G1284

Statement:

Use only one font for **HTML** body text.

Rationale:

Users may not have a wide variety of fonts available in their browser, so it is best to use a single, common font. The general standard is to make body text sans serif since most people find sans serif fonts easier to read on monitors and **serif** fonts better for printed materials.

Referenced By:

Human Factor Considerations for Web-Based User Interfaces
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1284.1]

Does the HTML or style sheet refrain from using more than one font?

Procedure:

Search all Web pages and style sheets for the word "font." Make sure only one type of font is used for body text. May need to visually inspect Web pages to see if a defined font style is used within the body.

Example:

Correct usage:

Cascading style sheet:

```
body.main {
   font:sans-serif;
}
```

HTML:

Incorrect usage:

Several font styles are used within a body.

G1285

Statement:

Use **relative font sizes**.

Rationale:

Relative font sizes make Web sites more accessible and support meeting the requirements of Section 508 of the Rehabilitation Act of 1973. Relative font sizes allow for a low-vision user to enlarge the size of the text. Relative font sizes also support maintainability by not hard coding fixed **font sizes**.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Human-Computer Interaction
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Interoperability

Evaluation Criteria:

1) Test: [G1285.1]

Are any absolute font sizes utilized?

Procedure:

Search all Web pages and style sheets for the word "font." Inspect the results to make sure no fixed fonts are used (e.g., 12pt).

Example:

Correct Usage

Relative or no font sizes settings are used.
Cascading style sheets:

```
p {
   font-size:200%;
}
p {
   font-size:2em;
}
```

Incorrect Usage

Cascading style sheets:

```
p {
   font-size:12pt;
}
```

HTML (the font attribute should not be used at all within HTML code, only external style sheets):

G1286

Statement:

Provide text labels for all buttons.

Rationale:

Users need to understand the purpose of all buttons. In some cases an image on the button is not sufficient to convey meaning. Screen scrapers used by the visually impaired work better when text labels are available for buttons

In cases where icons serve as buttons in order to fit within a small display device (such as a personal digital assistant), providing an option to enable text labels (or providing alternate attributes in the case of Web-based interfaces) supports screen scrapers.

Referenced By:

Interoperability
Human-Computer Interaction
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1286.1]

Do all buttons have associated text labels?

Procedure:

Inspect the user interface to verify text labels are available for all buttons.
Text labels may optionally be displayed:
- on or near the button
- as a tooltip when the user hovers over a button
- as part of a help system where a user clicks and identify tool and then clicks a button.

Button label text may not be enabled by default on all applications, especially systems with small resolution screens such as PDAs.

Example:

Correct usage:

```
<form action="mailto:me@abc.com"
method="post">
<input type="submit" name="emailbut"
value="Send feedback" />
</form>
```

Incorrect usage (using images only):

```
<input type="image" src="send.gif" name="
emailbut"/>
```

G1287

Statement:

Provide feedback when a transaction will require the user to wait.

Rationale:

Users may think that the application has stopped running or is malfunctioning.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Design Tenet: Enterprise Service Management
Human-Computer Interaction

Evaluation Criteria:

1) Test: [G1287.1]

Does the application provide feedback during long processes?

Procedure:

Run the application and observe any processes that take longer than 10 seconds to complete. Observe if any status indication is provided to alert the user of the status.

Example:

None

G1292

Statement:

Use text-based Web site navigation.

Rationale:

Text-based navigation works better than image-based navigation because it enables users to understand the link destinations. Users with text-only browsers and browsers with deactivated graphics can see only text-based navigation options.

Referenced By:

Design Tenet: Accommodate Heterogeneity
Human Factor Considerations for Web-Based User Interfaces
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability

Evaluation Criteria:

1) Test: [G1292.1]

Are there any instances where graphics are used for navigation?

Procedure:

Visually inspect all Web pages and make sure navigation elements are textual.

Example:

None

G1293

Statement:

Use descriptive labels for all clickable graphics.

Rationale:

Clickable images generally confuse users, especially images that contain only graphics. Some that contain both graphics and words are also confusing because users do not know if the images are clickable without using the mouse pointer.

Referenced By:

Human Factor Considerations for Web-Based User Interfaces
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1293.1]

Do Web pages contain clickable images?

Procedure:

Search all Web pages for image ("img") tags embedded inside link ("a") tags. Visually inspect each image found in the search and make sure there is an associated text description.

Example:

Correct Usage

```
Click myimage to go to www.mywebsite.com
<a href="www.mywebsite.com"><img src="myimage.gif"></a>
```

Incorrect Usage

```
<A href="www.mywebsite.com"><img
src="myimage.gif"></a>
```

G1294

Statement:

Provide a site map on all Web sites.

Rationale:

A site map shows explicit organization of the site. Inexperienced users do not readily form a mental model of the way that information is organized in a Web site, making it hard for them to recover from navigational errors.

Referenced By:

Human Factor Considerations for Web-Based User Interfaces
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability

Evaluation Criteria:

1) Test: [G1294.1]

Does the Web site have a site map?

Procedure:

Search all Web pages for anything with the name "sitemap," "site map" and "map." Visually inspect the search results to make sure a site map is included.

Example:

None

G1295

Statement:

Provide redundant text links for images within an **HTML** page.

Rationale:

Redundant text links for images within an **HTML** page allow users to navigate the **Web page** even if their browsers do not display images (as in situations where the **Web browser** renders content without images due to bandwidth considerations). Screen scrapers that assist the visually impaired also use redundant text links. Images may occur within Web pages as part of the content or navigation controls to include **image maps**.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity
Human Factor Considerations for Web-Based User Interfaces
Interoperability

Evaluation Criteria:

1) Test: [G1295.1]

Are alternative text links provided for all HTML page images used for navigation?

Procedure:

Verify that alternative text links are provided for images used for navigation by inspecting the HTML source code and testing the HTML page in a browser with image rendering turned off.

Example:

None.

G1300

Statement:

Secure all **endpoints**.

Rationale:

Something is only as secure as its weakest link. Therefore, all access points in an application should be secured. An endpoint is defined as an entry or an exit point of an application. Any access point can be vulnerable to attacks. For instance, if an application file reads configuration settings from a properties file, that file can be corrupted or incorrectly configured. This can cause incorrect behavior in the application. Also if component, **module** or application provides remote access or is part of any inter-process communications, these areas are vulnerable to attacks. For instance, if the application provides an external socket interface, does it validate commands being sent by the client?

Referenced By:

Interoperability
General Application Security
Design Tenet: Identity Management, Authentication, and Privileges
Maintainability

Evaluation Criteria:

1) Test: [G1300.2]

Does the application handle invalid configuration, provide appropriate defaults, and protect sensitive data?

Procedure:

Check application processing of data files (configuration files, properties files, preferences, XML, etc.).

Example:

None.

2) Test: [G1300.1]

Does the application properly handle security when dealing with externally accessible API(s) and external ports?

Procedure:

Verify sensitive data is protected, and verify all network base protocols validate commands and values.

Example:

None.

G1301

Statement:

Practice layered security.

Rationale:

An application with layered security provides more protection against attacks. Combining multiple layers of security defenses can provide additional protection when one layer is broken.

Referenced By:

General Application Security
Other Design Tenets
Interoperability
Maintainability
Practice Defense in Depth
Design Tenet: Layering and Modularity

Evaluation Criteria:

1) Test: [G1301.1]

Do internal and external API(s) perform security checks?

Procedure:

Make sure layers of API(s) starting from externally accessible API(s) down through the layers of internally accessible API(s) provide sufficient security checks. For example, does each layer of the API perform data validation? If internal API is calling remote services, is the data sufficiently protected from snoopers (e.g., use of secure sockets)?

Example:

None

2) Test: [G1301.2]

Does the application handle security when processing data files?

Procedure:

Embed all application specific resources such as graphics, internal application configuration files such as internationalization properties/resources, XML files as part of a signed application deployment file (.jar, .exe, etc.).

Example:

None

G1302

Statement:

Validate all inputs.

Rationale:

Do not limit input validation to the presentation tier; rather, all external APIs should validate inputs prior to use. This is just one aspect of defense in depth which can prevent many attacks including SQL Injection, Cross-Site Scripting, Buffer Overflows, and Denial of Service.

Referenced By:

Other Design Tenets
Validate Input
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
General Application Security

Evaluation Criteria:

1) Test: [G1302.2]

Does the application provide proper handling for null input?

Procedure:

Check application handling of null values.

Example:

None

2) Test: [G1302.1]

Does the application use prefix or postfix validation (asserts) to verify input parameters?

Procedure:

Check application range validation of externally accessible API(s).

Example:

None

G1304

Statement:

Unit test all code.

Rationale:

A high percentage of all security violations can be attributed to inadequate or non-existent unit testing. Hackers can take advantage of these.

Referenced By:

General Application Security
Interoperability
Apply Quality Assurance to Software Development

Evaluation Criteria:

1) Test: [G1304.1]

Does the project unit test the code base?

Procedure:

Use a coverage tool to determine how much of the project's code have been tested.

Check for use of a unit testing framework (JUnit for example).

Example:

None

G1305

Statement:

Ensure the separation of **encrypted** and unencrypted information.

Rationale:

Not separating encrypted and unencrypted information can cause the application to incur performance hits due to unnecessary encryption. It can also cause inconsistent application processing.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Encryption and HAIPE
Other Design Tenets
Interoperability
General Application Security

Evaluation Criteria:

1) Test: [G1305.1]

Does the data model separate sensitive data from other data?

Procedure:

Check **UML** or entity diagram to ensure that separate components or entities are used to defined sensitive data.

If annotation support is provided via **XML**, ensure that the data is properly labeled (XML attribute) with correct security attributes.

Example:

None

G1306

Statement:

Identify and **authenticate** users of the application.

Rationale:

This ensure there is some traceability and also provides the first in a multilayer security system.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
General Application Security

Evaluation Criteria:

1) Test: [G1306.2]

Does the application authenticate with another service (**LDAP**, database or simple password)?

Procedure:

Inspect application code to ensure that the user is authenticated against an LDAP, database or simple password service.

Example:

None

2) Test: [G1306.1]

Does the application require user certificates?

Procedure:

Ensure the application is setup to require client side certificates. This can be done easily by using a machine without any DoD client certificates installed and attempting to access the application.

Example:

None

G1307

Statement:

Provide a security policy file.

Rationale:

Security should not be an afterthought after application design and implementation. A security policy file can go along way in ensuring that application security has been part of the design and implementation of the application. A security policy file can identify all the security measures that the application has laid out.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Maintainability
General Application Security

Evaluation Criteria:

1) Test: [G1307.1]

Does the project have Security Policy File?

Procedure:

Check for the existence of a Security Policy file.

Example:

None

G1308

Statement:

Configure **Public Key Enabled** applications to use a **Federal Information Processing Standard** (**FIPS**) 140-2 certified cryptographic module.

Rationale:

The guidance defines the application types required to support DoD class 3 PKI.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Maintainability
Interoperability
Public Key Infrastructure (PKI) and PK Enable Applications
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1308.1]

Is the application using an approved **Federal Information Processing Standard** (**FIPS**) 140-1 cryptographic **module**?

Procedure:

Check the cryptographic module to see if it is FIPS 140-2 compliant.

Example:

None

G1309

Statement:

Make applications handling high value unclassified information in Minimally Protected environments **Public Key Enabled** to interoperate with **DoD High Assurance** .

Rationale:

This guidance defines the application types required to support DoD High Assurance (Mission Assurance Category I [MAC I]) certificates.
The definition of MAC I is "systems handling information that is determined to be vital to the operational readiness or mission effectiveness of deployed and contingency forces in terms of both content and timeliness. The consequences of loss of integrity or availability of a MAC I system are unacceptable and could include the immediate and sustained loss of mission effectiveness. MAC I systems require the most stringent protection measures." (DoD Instruction 8580.1, ***Information Assurance (IA) in the Defense Acquisition System***, 9 July 2004. [R1199])

Note: *This guidance is derived from DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* *[R1206]*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Public Key Infrastructure (PKI) and PK Enable Applications
Maintainability

Evaluation Criteria:

1) Test: [G1309.1]

Is the application using a High Assurance key material generated in a **Federal Information Processing Standard** (**FIPS**) 140 Level 2 validated hardware cryptographic **module**?

Procedure:

Check cryptographic module to see if it is FIPS 140 Level 2 compliant.

Example:

None.

G1310

Statement:

Protect application cryptographic objects and functions from tampering.

Rationale:

If cryptographic objects such as private keys, key store, and CA trusted certificates are not protected, the system is not secure.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Public Key Infrastructure (PKI) and PK Enable Applications
Interoperability

Evaluation Criteria:

1) Test: [G1310.1]

Are cryptographic objects protected?

Procedure:

Check that key stores, private keys, and **trust points** are protected.

Verify a documented procedure for creating and documenting the creation of keys exists.
Verify a documented procedure for obtaining certificates exists.
Verify a documented procedure for backing up cryptographic objects exists.

Example:

Use High Security Level setting in Internet Explorer to ensure password protection is used. See https://infosec.navy.mil/PKI/certs.html for software certificate steps. See https://infosec.navy.mil/PKI/cac.html for CAC.

G1311

Statement:

Use **Hypertext Transfer Protocol over Secure Socket Layer** (**HTTPS**) when applications communicate with DoD **Public Key Infrastructure** (**PKI**) components.

Rationale:

These are the DoD approved protocols and the only supported ones.

Note: *This guidance is derived from DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* *[R1206]*

Referenced By:

Public Key Infrastructure (PKI) and PK Enable Applications
Interoperability
Reusability
Maintainability
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1311.1]

Does the application use only HTTPS to communicate when using DoD PKI?

Procedure:

Have application access the DoD PKI Global Directory Service (GDS) Directory (dod411.gds.disa.mil/) via HTTPS.

Example:

None

G1312

Statement:

Make applications capable of being configured for use with DoD **PKI**.

Rationale:

Applications must be configurable to request and install certificates, add **trust points**, and require client authentication.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.4, Version 1.0, 13 July 2000.*

Referenced By:

Interoperability
Design Tenet: Identity Management, Authentication, and Privileges
Public Key Infrastructure (PKI) and PK Enable Applications
Maintainability

Evaluation Criteria:

1) Test: [G1312.1]

Is there a capability to configure the application for use with DoD PKI?

Procedure:

Check to make sure the application is configurable to accept certificates, load key stores, and add **trust points**; this may involve inspecting user and administrator manuals.

Example:

None

G1313

Statement:

Provide documentation for application configuration and setup for use with DoD **PKI**.

Rationale:

If the application can not be configured or setup correctly, the application is insecure. Without detail documentation, personnel with little knowledge of security or PKI will have little chance of keeping the overall system secure. The Navy Public Key Infrastructure training site, https://infosec.navy.mil/PKI/training.html (DoD PKI Certificate required for access), contains links to several configuration guides.

Note: *This guidance is derived from the DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* *[R1206]*

Referenced By:

Maintainability
Public Key Infrastructure (PKI) and PK Enable Applications
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1313.1]

Is there documentation (such as Standard Operating Procedures [SOPs]) on how to configure and setup the application to interoperate within the DoD PKI?

Procedure:

Verify by inspection of the SOPs and by a demonstration that the application performs as documented when the configuration guidance is followed.

Example:

Most application manuals have detailed instructions in enabling PKI (either under the heading "enabling SSL" or "certificates").

G1314

Statement:

Provide applications the ability to import and export keys (software certificates only).

Rationale:

The whole PKI system is predicated on the use of public-private key pair. The ability to import and use private keys is critical to a functional PKI application.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.5, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Key Management
Interoperability
Maintainability

Evaluation Criteria:

1) Test: [G1314.1]

Is the application able to import and export keys associated with standard certificates for individuals?

Procedure:

Have the application import and export at least one set of keys and certificates for each certificate type supported by the application. Demonstrate interoperability by performing representative subscriber and relying party operations with each certificate type and its related keys.

Note: *Verify the correctness of the exported file through analysis.*

Example:

Internet Explorer can import/export certificates using Tools > Internet Options. Click on Internet tab and then click on Certificates link. Import/Export options are located here.

UNIX-based Web server keys are exported by making a copy of the keys file and placing it in a safe location.

G1315

Statement:

For applications, use key pairs and **Certificates** created for individuals using DoD **PKI** methods and procedures defined by the DoD Class 3 Public Key Infrastructure Interface Specification and the Personal Information Exchange Syntax Standard.

Rationale:

DoD PKI supports these standards for importing keys and certificates. If the key or certificate is not created or issued by approved DoD Certificate architecture, it can not be trusted to interoperate within the DoD network.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.5, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Maintainability
Interoperability
Key Management

Evaluation Criteria:

1) Test: [G1315.1]

Can the application import and export keys associated with standard certificates for individuals?

Procedure:

Verify by importing and exporting to DoD PKI key store.

Access the application using a DoD PKI Class 3 Certificate.

Example:

For servers, verify that the application requires client side authentication. Access the application server using a DoD PKI certificate.

G1316

Statement:

Ensure that applications protect **private keys**.

Rationale:

In order for the PKI system to stay secure, the private key must not be compromised. Protecting the private key helps prevent attackers from decrypting secured data communications.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.5, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Key Management

Evaluation Criteria:

1) Test: [G1316.1]

Does the application use and store the private key securely?

Procedure:

Check for the following:

- all copies of the private key destroyed when private key operation is complete; for example, check that the private key does not stay in application memory permanently

- the private key is password protected with a strong password
- the **keystore** is password protected with a strong password

Example:

Attempt to view the contents of the private key using a document viewer program.

G1317

Statement:

Ensure applications store **Certificates** for subscribers (the owner of the **Public Key** contained in the Certificate) when used in the context of signed and/or encrypted email.

Rationale:

This will allow other parties to use the public key to encrypt messages sent to the application.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document. Section (4.5), Version 1.0, July 13, 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Key Management
Interoperability

Evaluation Criteria:

1) Test: [G1317.1]

Is the public key available from the Directory Server application?

Procedure:

See if it is possible to extract the public key certificate from the Directory Server application.

Example:

None

G1318

Statement:

Develop applications such that they provide the capability to manage and store **trust points** (**Certificate Authority** Public Key **Certificates**).

Rationale:

This will ensure the certificate is valid and expedite verification of the certificate.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Key Management
Interoperability
Maintainability

Evaluation Criteria:

1) Test: [G1318.1]

Is the Certificate Authority public key available from the application?

Procedure:

View the application's trust list to verify DoD PKI Class 3 CA certificates are present.

Example:

For Internet Explorer, view the DoD PKI Class 3 CA certificates by selecting **`Tools`**>**`Internet Options`**. Click on the **`Internet`** tab and then click on the **`Publishers`** button. Click on the **`Trusted Root Certification Authorities`** tab and scroll down to verify that the DoD PKI Class 3 CA certificates are present.

Web server Certificate Authority certificates can usually be viewed by the application's GUI. If a GUI is not offered, reference the application's manual concerning certificate management.

G1319

Statement:

Ensure applications can recover data encrypted with legacy keys provided by the DoD **PKI** Key Recovery Manager (**KRM**).

Rationale:

Applications may have the need to decrypt legacy information that the application originally encrypted.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Key Management
Interoperability
Maintainability

Evaluation Criteria:

1) Test: [G1319.1]

Is the application able to recover legacy encrypted data?

Procedure:

Acquire the legacy key and demonstrate the ability
to decrypt data that is encoded by that key.

Example:

None

G1320

Statement:

Use a minimum of 128 bits for **symmetric keys**.

Rationale:

Strong encryption helps to prevent unauthorized data decryption using modern day resources.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Design Tenet: Encryption and HAIPE
Interoperability
Maintainability
Encryption Services

Evaluation Criteria:

1) Test: [G1320.1]

Are symmetric key encryption levels at least 128 bit?

Procedure:

Check the server configuration and verify that the symmetric keys being used are at least 128 bit.

Example:

Verified Web server ciphers under the SSL portion of the configuration pages of the administration server.

For Internet Explorer 5.0 and above, click the **`Help`** menu and then click the **`About Internet Explorer`** option. The About box will list the Cipher Strength.

2) Test: [G1320.2]

Is the application using domestic (U.S.) grade ciphers?

Procedure:

Verify that the application supports domestic (U.S.) grade ciphers.

Example:

None.

G1321

Statement:

Enable applications to be capable of performing **Public Key** operations necessary to verify signatures on DoD **PKI** signed objects.

Rationale:

An application must verify the digital signature and check its validity against the current **Certificate Revocation List** (**CRL**) maintained by an on-line repository (e.g., **Online Status Check Responder** or **OSCR**).

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Encryption and HAIPE
Maintainability
Design Tenet: Identity Management, Authentication, and Privileges
Encryption Services
Reusability
Interoperability

Evaluation Criteria:

1) Test: [G1321.1]

Does the application verify signed objects?

Procedure:

Check that the application validates signed objects against DoD root certificates.

Check that the signing certificate has not been revoked by checking against Certificate Revocation Lists or using the Online Certificate Status Protocol (OCSP).

Example:

Make a back-up copy of the certificate. For Windows based applications, stop the application and edit the signature of the certificate and save the certificate. Start the application back up. The application should fail to start as the signature check will fail.

For validity checking, confirm a validity check of the certificate was performed by viewing the application's audit log.

G1322

Statement:

Ensure that applications that interact with the DoD **PKI** using **SSL** (i.e., **HTTPS**) are capable of encrypting and decrypting data using the **Triple Data Encryption Algorithm** (**TDEA**).

Rationale:

Applications should use cryptographic modules approved under **Federal Information Processing Standard** (**FIPS**) 140, Level 1.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Maintainability
Design Tenet: Encryption and HAIPE
Design Tenet: Mediate Security Assertions
Encryption Services
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability

Evaluation Criteria:

1) Test: [G1322.1]

Does the application use TDEA for encrypting and decrypting data?

Procedure:

Inspect the application's configuration file to confirm that TDEA is used for encrypting and decrypting data.

Example:

Most server based applications have cipher related information stored under SSL, certificates, or security. Verify that the application is using TDEA.

G1323

Statement:

Generate random **symmetric encryption** keys when using symmetric encryption.

Rationale:

If the application can not generate random keys, then it is vulnerable to attacks if attackers can determine the algorithm for generating the random symmetric encryption keys.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Maintainability
Design Tenet: Encryption and HAIPE
Encryption Services
Interoperability

Evaluation Criteria:

1) Test: [G1323.1]

Does the application generate random symmetric encryption keys?

Procedure:

Verify that the random seed is generated (e.g., by viewing the application's vendor documentation).

Example:

Most server based applications either user MOD_SSL or OPEN_SSL. These two toolkits properly use random seed generators.

Apache based servers may require the administrator to type random keystrokes on the keyboard. This process is generating the random seed.

G1324

Statement:

Protect **symmetric keys** for the life of their use.

Rationale:

Symmetric key encryption algorithms are based on trivially related keys for both encryption and decryption. The advantage of symmetric key encryption is that it is much less computationally intensive for encryption and decryption compared to asymmetric algorithms. The disadvantage is that the shared symmetric key must be kept secure during storage and transmission.

To prevent disclosure, new symmetric keys are often generated for each unique **session** and exchanged using another encryption algorithm. Store symmetric keys that are used long term carefully to prevent disclosure.

> ***Note:*** *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Encryption Services
Interoperability
Design Tenet: Encryption and HAIPE
Maintainability

Evaluation Criteria:

1) Test: [G1324.1]

Are symmetric keys stored in unprotected locations?

Procedure:

Check for hard coded symmetric keys in source code or files with weak permissions.

Example:

Symmetric keys should be generated for each session and destroyed when the session is destroyed, never stored in a file with weak permissions or hard coded in source code.

G1325

Statement:

Encrypt **symmetric keys** when not in use.

Rationale:

Symmetric keys enable both sides of the conversation to have knowledge of the key for encryption. It can not be given out freely, which means if it is going to be stored for repeated use, it should be encrypted first before storage.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Maintainability
Encryption Services
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1325.1]

Does the application encrypt symmetric keys when not in use?

Procedure:

Check that the application encrypts symmetric keys during storage.

Example:

None.

G1326

Statement:

Ensure applications are capable of producing Secure Hash Algorithm (**SHA**) **digests** of **messages** to support verification of DoD **PKI** signed objects.

Rationale:

Symmetric keys enable both sides of the conversation to have knowledge of the key for encryption. It can not be given out freely, which means if it is going to be stored for repeated use, it should be encrypted first before storage.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Interoperability
Design Tenet: Encryption and HAIPE
Maintainability
Encryption Services
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1326.1]

Does the application use SHA digest?

Procedure:

Visually validate that the SHA digest is used for symmetric keys.

Example:

Most application servers allow one to configure the hash to SHA1. Please note that the default for most applications is MD5.

G1327

Statement:

Enable an application to obtain new **Certificates** for subscribers.

Rationale:

If the application generates subscriber keys, the application shall demonstrate the ability to generate keys, request new certificates, and obtain new certificates through interaction with the DoD PKI. If the generated keys are for encryption applications, the application shall demonstrate its ability to provide keys to the DoD PKI KRM.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.3.2.2, Version 1.0, 13 July 2000.*

Referenced By:

Certificate Processing
Maintainability
Interoperability
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1327.1]

Can the application request and obtain new certificates for subscribers?

Procedure:

For application servers, verify that the application can successfully request a certificate via the appropriate certificate request page from a DoD PKI CA.

For application servers, verify that the application can successfully download an issued certificate from a DoD PKI CA.

Example:

Instructions in obtaining a DoD PKI certificate for a user are available at https://infosec.navy.mil/PKI/users.html.

Instructions for obtaining a DoD PKI certificate for web servers including Netscape, Lotus, and IIS is available at https://infosec.navy.mil/PKI/training.html.

G1328

Statement:

Enable an application to retrieve **Certificates** for use, including relying party operations.

Rationale:

The ability to retrieve certificates from DoD certificate repositories further ensures the authenticity of the certificate .

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.3.2.3, Version 1.0, 13 July 2000.*

Referenced By:

Interoperability
Certificate Processing
Maintainability
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1328.1]

Can the application retrieve **Certificates** from a DoD PKI certificate repository?

Procedure:

Verify that the application can communicate with a DoD PKI certificate repository such as GDS.

Example:

This test procedure is only required for applications that must send encrypted e-mail. For this scenario, assume that Outlook is used; instructions for using Outlook 2000 are available at https://infosec.navy.mil/PKI/Outlook_2000_0704.pdf

G1330

Statement:

Ensure applications are capable of checking the status of **Certificates** using a **Certificate Revocation List** (**CRL**) if not able to use the **Online Certificate Status Protocol** (**OCSP**).

Rationale:

Applications must verify the validity of the certificate prior to establishing trust with another entity. **CRL** is the legacy mechanism for validating certificates. Applications should favor **OSCP** for new development.

Applications operating in environments with network connectivity to a**CRL distribution point** should be able to obtain a current CRL. Applications should be able, without user intervention, to obtain a current CRL to check the status of a certificate that contains a CRL distribution point extension. Applications with network connectivity unable to find CRL distribution points automatically should be capable of being configured with a distribution point that the application then uses to obtain CRLs as needed.

Systems on DoD networks must use a local Web cache to obtain the latest DoD PKI issued CRL per Joint Task Force Global Network Operations (JTF GNO) Communications Tasking Order (CTO) 07-015 of 11 December 2007 (specifically Task 11; DoD PKI Certificate required for access). Configuration instructions for known Web cache products in use and alternative CRL caching capabilities are available from the following location: https://www.us.army.mil/suite/page/474113 (Army or Defense On Line [AKO or DKO] site registration and DoD PKI Certificate required for access).

Note: *This guidance is derived from DoD Instruction 8520.2,* ***Public Key Infrastructure (PKI) and Public Key (PK) Enabling****, 1 April 2004.* *[R1206]*

Referenced By:

Design Tenet: Network Connectivity
Certificate Processing
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Maintainability

Evaluation Criteria:

1) Test: [G1330.1]

Can the application perform Certificate status checking with a CRL?

Procedure:

Verify that the application can download a CRL successfully .

Example:

Visually inspect the application is configured to use CRLs for validity checking. This can be achieved by looking at the directory in which the application stores the CRLs.

G1331

Statement:

Ensure applications are able to check the status of a Certificate using the **Online Certificate Status Protocol** (**OCSP**).

Rationale:

Applications must verify the validity of the certificate prior to establishing trust with another entity. CRL is the legacy mechanism for validating certificates. Applications should favor **OCSP** for new development.

Applications may use an OSC responder to check the status of a particular certificate when the DoD has an operational responder. Applications shall prepare and transmit the request to the responder using HTTP in accordance with the DoD Class 3 PKI Infrastructure Interface Specification.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Section 4.3.2.4.2, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Maintainability
Certificate Processing

Evaluation Criteria:

1) Test: [G1331.1]

Can the application perform **Certificate** status checking with **OCSP**?

Procedure:

Verify that the application can performing OCSP queries to an **OSC** Responder successfully.

Example:

Visually inspect the application is configured to use OCSP for validity checking. This can be achieved by looking at the configuration file to see that the application is configured to use OCSP. One can also visually look at the application's log file to validate that the application is making OCSP queries.

G1333

Statement:

Only use a **Certificate** during the Certificate's validity range, as bounded by the Certificate's "Validity - Not Before" and "Validity - Not After" date fields.

Rationale:

Expired certificates should not be accepted except in cases where legacy data was archived.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Certificate Processing
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Maintainability

Evaluation Criteria:

1) Test: [G1333.1]

Do the date and time of the use of the Certificate fall within the Certificate's validity period?

Procedure:

Visually inspect the certificate's validity dates. The certificate should be valid and not expired.

Example:

Each digital certificate has a lifetime. When viewing a certificate, the certificate will have a valid from date and a valid to date. The current date should fall within this range.

G1335

Statement:

Make applications capable of being configured to operate only with PKI Certificate Authorities specifically approved by the application's owner/managing entity.

Rationale:

Using approved PKI Certificate Authorities ensures certificate authenticity and ensures that the certificate is chained to the issuer.DoD trust points ensure certificates are chained to the issuer of the certificate and are authentic.

For example, DoD applications are configured to use DoD PKI Certificate Authorities only per the DoD Class 3 PKI - Public Key-Enabled Application Requirments Document Version 1.0, 13 July 2000.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Certificate Processing
Reusability
Maintainability

Evaluation Criteria:

1) Test: [G1335.1]

Is the application configured to operate only with approved PKI Certificate Authorities?

Procedure:

Visually inspect that only the DoD PKI certificates are trusted by the application.

Example:

Applications typically allow one to view the trust points via the administrative interface to the application. CA certificates are typically located under Certificate Management, SSL, or Security.

G1338

Statement:

Applications and **Certificates** need to be able to support multiple organizational units.

Rationale:

DoD requirements dictate that certificates shall support multiple organizational units.

Note: *This guidance is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

Referenced By:

Maintainability
Certificate Processing
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability

Evaluation Criteria:

1) Test: [G1338.1]

Can the application process a **Certificate** that contains multiple organizational units in the Distinguished Name?

Procedure:

Visually inspect the DoD PKI CA certificates stored in the application. You will notice that each certificate contains multiple organizational units (OU=DoD, OU=PKI)

Example:

The majority of certificate request forms do not contain entries for multiple organizational units. In this case, include all of the organizational unit information in the single line. For example, for Navy, please enter the following information next to the Organizational Unit line: Navy, OU=DoD, OU=PKI.

Once the certificate is issued, visually inspect this certificate to verify that the certificate contains these Organizational Unit values.

G1339

Statement:

Practice defensive programming by checking all method arguments.

Rationale:

Data validation is not limited to Graphical User Interfaces. API(s) and library functions are also susceptible to corruption. The integrity of application can benefit from identifying invalid data as early as possible.

Referenced By:

Validate Input
Interoperability
API Security
Other Design Tenets

Evaluation Criteria:

1) Test: [G1339.1]

Does the application perform range validation?

Procedure:

Check for unit tests.

Check thrown exceptions.

Purposely send invalid data to API(s) to test the integrity and handling of invalid data.

Example:

None.

G1340

Statement:

Log all exceptional conditions.

Rationale:

Logging exceptional conditions can help to identify security problems, trace the source of the exception, and trigger security alerts.

Referenced By:

API Security
Maintainability
Handle Exceptions
Other Design Tenets

Evaluation Criteria:

1) Test: [G1340.1]

Does the application perform logging of exceptional conditions?

Procedure:

Check exception handlers for logging support.

Example:

None.

G1341

Statement:

Use a security manager support to restrict application access to privileged system resources.

Rationale:

Desktop applications by default do not install a security manager. Installing a security manager could prevent unsecured access to system resources such as network and file system. Desktop applications can benefit from using a security manager to ensure that system resources are protected.

Referenced By:

Java Security
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Design Tenet: Cross-Security-Domains Exchange

Evaluation Criteria:

1) Test: [G1341.1]

Does an installed security manager restrict application access to privileged system resources?

Procedure:

Check application main method for installation of a security manager.

Example:

None.

G1342

Statement:

Restrict direct access to class internal variables to functions or methods of the class itself.

Rationale:

One of the primary tenets in Object Oriented Programming is encapsulation. Restricting access to internal variables not only secure the Class/Object against corruption (no data validation), it is also a maintenance issue. Hiding the implementation details allows the flexibility of underlying implementation to change.

Referenced By:

Maintainability
Java Security
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1342.1]

Do classes directly expose internal data members?

Procedure:

Make sure all internal class variables are declared private or protected.

Example:

None.

G1343

Statement:

Declare classes final to stop inheritance and prevent methods from being overridden.

Rationale:

Utility classes and classes that do not intend to be extended (classes used for user authentication) should lock down their implementation. Locking implementation can prevent methods from being overridden. Not locking down implementation can cause corruption of internal class data or allow errant code to run. For example, imagine the possibility of a class that performs credit card processing that can be overridden.

Class implementation can be locked down by declaring the class or methods final.

Referenced By:

Interoperability
Maintainability
Java Security

Evaluation Criteria:

1) Test: [G1343.1]

Are sensitive, security related, and utility classes declared final?

Procedure:

Check classes used in Security related processing (authentication, authorization) final keyword.

Check classes that have sensitive data (social security numbers, medical data, and salary information) for final keyword.

Check Utility classes for final keyword.

Example:

None.

G1344

Statement:

Encrypt sensitive data stored in configuration or resource files.

Rationale:

Sensitive data used for application configuration files (XML), user profiles, or resource files should be protected from tampering. The sensitive data should be encrypted and or a message **digest** or checksum should be calculated to check for tampering. Application should handle generation, accessing and storing data to these files.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Application Resource Security
Interoperability
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1344.1]

Is sensitive data in configuration files and user profiles?

Procedure:

Check properties files, XML configuration files or user profiles for sensitive data in the clear.

Check for an application to edit, and creation of the file.

Example:

None.

G1346

Statement:

Audit database access.

Rationale:

Auditing is critical for data access traceability. If the RDBMS was attacked, auditing is essential not only for figuring out what had occurred but also to recover lost data. Database access auditing provides logs for each access or change to the database by a given user (or an IP address for systems without user authentication).

Often current middle tier technologies (e.g., J2EE, .Net, CORBA, etc.) share database connections and may only have a single database user. Thus the burden is on the middle tier to know the identity of each user and be able to pass this information on the database (e.g., design each table to have data items such as updated by, created by, etc.).

Referenced By:

RDBMS Security
Other Design Tenets
Design Tenet: Identity Management, Authentication, and Privileges
Maintainability

Evaluation Criteria:

1) Test: [G1346.1]

Does the application database include actual user rather than database connection owner?

Procedure:

Check system documentation, database tables, and audit logs to verify that database access audit entries are created for each database access.

Example:

None

G1347

Statement:

Secure remote connections to a database.

Rationale:

Just because the database is behind the corporate firewall does not mean someone inside the firewall cannot access or listen in on the wire.

Net-centricity implies that a database should be on the network and not constrained to be sitting behind an application server. This means that many unanticipated users may eventually access the database. Thus, database security should not be based on isolation.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
RDBMS Security
Interoperability
Design Tenet: Decentralized Operations and Management

Evaluation Criteria:

1) Test: [G1347.1]

Is data exchanged between the database and client secure?

Procedure:

Check for secure protocol (e.g., SSL) between application and database.

Check for secure data access by IP address.

Check for configuration in the database (user) which limits user from a specified host.

Example:

None.

G1348

Statement:

Log database **transactions**.

Rationale:

Transaction logging is generally handled by the database management system and records all changes made to the database, critical for data recovery and traceability.

Referenced By:

Maintainability
Other Design Tenets
RDBMS Security

Evaluation Criteria:

1) Test: [G1348.1]

Are database transactions logged?

Procedure:

Commercial database management systems have a feature to log database transactions. Check to determine whether the feature has been turned on in the database management system.

Example:

None.

G1349

Statement:

Validate all input that will be part of any dynamically generated **SQL**.

Rationale:

Not validating or filtering parameters used in dynamically generated SQL statements can lead to SQL injection attacks.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
RDBMS Security
Other Design Tenets
Interoperability
Validate Input

Evaluation Criteria:

1) Test: [G1349.1]

Does the database use filtering or data validation code?

Procedure:

Filter out character like single quote, double quote, slash, back slash, semi colon, extended character like NULL, carry return, new line, etc, in all input strings.

Example:

G1350

Statement:

Implement a strong password policy for **RDBMS**.

Rationale:

Clean database installation often contains no passwords for root users. Also, new user accounts often defaults to no password or standard password. Having no passwords allows users access any data. Database users should always be given strong passwords. This implies a non null password, locking unused user accounts and ensuring that system user accounts are not using default passwords

Referenced By:

RDBMS Security
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability

Evaluation Criteria:

1) Test: [G1350.1]

Does the database user table include passwords?

Procedure:

Check for null or empty values for passwords in the user table.

Use a commercially available or open source default password analysis tool to ensure that all user accounts do not retain default passwords and to ensure that all passwords are strong.

Example:

None.

G1351

Statement:

Enhance database security by using multiple user accounts with constraints.

Rationale:

Constrain access to individual tables and functions by creating multiple user accounts for an application and constraining the accounts to specific functions. As a general policy, user accounts should be constrained to the minimal required database access. For example, creation of a read only account should be constrained by granting only select on the tables of interest to the read only user. This aids in password management as well as limiting the potential impact of SQL injection attacks. By granting only insert on a table, for example, and not granting select, the user could in effect create a write only database.

Each application will have different requirements in regards to grants and access to tables. If one application is compromised, it will not affect the other applications.

It also has traceability to determine which application has allowed a security violation.

Referenced By:

Interoperability
Design Tenet: Identity Management, Authentication, and Privileges
RDBMS Security

Evaluation Criteria:

1) Test: [G1351.1]

Does each database application user have account constraints in accordance with the user function?

Procedure:

Check each database application user to ensure that the account constraints are in accordance with the user function and do not have unwarranted privileges. For example, check that read only application user accounts have only read access enabled.

Example:

None.

G1352

Statement:

Use database clustering and redundant array of independent disks (RAID) for high availability of data.

Rationale:

Database clusters combined with RAID technology (e.g., data striping and mirroring) can help ensure continued operation of a system that suffers hardware or software failure.

Referenced By:

RDBMS Security
Design Tenet: Availability
Maintainability
Design Tenet: Scalability
Interoperability

Evaluation Criteria:

1) Test: [G1352.1]

Is the system designed to support high availability?

Procedure:

Check for the existence of a cluster and/or failover capability.

Check for the existence of RAID data storage for the database.

Example:

None.

G1356

Statement:

Use the **SOAP** standard for all **Web services**.

Rationale:

The Web services security specifications are designed as an extension of SOAP. The specs are unusable without SOAP.

Referenced By:

Reusability
XML Web Service Security
Design Tenet: Open Architecture
Interoperability
Maintainability
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1356.1]

Does the Web service user generate SOAP formatted XML messages?

Procedure:

Generate a test message and check it for SOAP compliance.

Example:

None.

2) Test: [G1356.2]

Does the Web service provider generate SOAP formatted XML?

Procedure:

Generate a test message and check it for SOAP compliance.

Example:

None.

G1357

Statement:

Do not rely solely on transport level security like **SSL** or **TLS**.

Rationale:

Web services inherently involve multiple intermediaries between the message sender and the ultimate destination. The intermediaries may not use transport level security. SSL and TLS do not provide end-to-end security, only security at the transport layer and only point-to-point. The use of SSL or TLS should depend on the needs of the system. For sensitive applications, augment the use of SSL/TLS with defense in depth measures such as message-level security mechanisms.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
XML Web Service Security
Design Tenet: Encryption and HAIPE
Design Tenet: Mediate Security Assertions

Evaluation Criteria:

1) Test: [G1357.1]

Does the Web service user generate encrypted XML messages?

Procedure:

Generate a test message and check it for encryption.

Example:

2) Test: [G1357.2]

Does the Web service provider generate encrypted XML messages?

Procedure:

Generate a test message and check it for encryption.

Example:

G1359

Statement:

Bind **SOAP Web service** security policy assertions to the service by expressing them in the associated **WSDL** file.

Rationale:

A Web service may be registered in zero, one, or multiple **UDDI** registries. By placing the security policy assertions in the Web service's WSDL file, they are readily available to all the consumers of the service regardless how the service was discovered

Referenced By:

XML Web Service Security
Design Tenet: Mediate Security Assertions
Interoperability
Maintainability
Other Design Tenets

Evaluation Criteria:

1) Test: [G1359.1]

Are Web service security policy assertions bound in the service WSDL file?

Procedure:

Check the Web Service's WSDL file for policy assertions.

Example:

None

G1362

Statement:

Validate incoming XML-based messages using a **schema**.

Rationale:

Prevent malicious agents from compromising the integrity of a service.

Referenced By:

XML Web Service Security
Design Tenet: Identity Management, Authentication, and Privileges
Validate Input
Interoperability

Evaluation Criteria:

1) Test: [G1362.1]

Does the Web service provider validate incoming messages?

Procedure:

Identify the existence of an XML Schema file and examine code to verify that all incoming messages are checked to be XML Valid.

Example:

None

G1363

Statement:

Do not use clear text passwords.

Rationale:

Prevent a hacker from intercepting and seeing a real password.

Referenced By:

XML Web Service Security
Design Tenet: Encryption and HAIPE
Interoperability
Design Tenet: Identity Management, Authentication, and Privileges
Other Design Tenets

Evaluation Criteria:

1) Test: [G1363.1]

Does the Web service user utilize a username/password token?

Procedure:

Generate a test message and check it for clear text passwords.

Example:

None

G1364

Statement:

Hash all passwords using the combination of a timestamp, a **nonce** and the password for each **message** transmission.

Rationale:

This Guidance helps to prevent unwanted interception or discovery of clear-text-hashed passwords.

Referenced By:

Design Tenet: Encryption and HAIPE
XML Web Service Security
Other Design Tenets
Design Tenet: Identity Management, Authentication, and Privileges
Interoperability

Evaluation Criteria:

1) Test: [G1364.1]

Does the Web service user utilize a username/password token?

Procedure:

Generate a test message and check it for a username/password token and verify that is contains a timestamp entry and a nonce entry.

Example:

None

G1365

Statement:

Specify an expiration value for all security tokens.

Rationale:

Specifying an expiration value for security tokens limits the chance of being able to intercept and use a security token to impersonate an authenticated user or process.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Other Design Tenets
XML Web Service Security

Evaluation Criteria:

1) Test: [G1365.1]

Does the Web service user utilize an expiration for each security token?

Procedure:

Generate a test message and check it to make sure an expiration is associated with each security token.

Example:

None

G1366

Statement:

Digitally sign all **messages** where non-repudiation is required.

Rationale:

Prevent hackers from changing intercepting and modifying a message.

Note: *Non-repudiation cannot be assured with soft certificates.*

Referenced By:

XML Web Service Security
Design Tenet: Identity Management, Authentication, and Privileges
Design Tenet: Encryption and HAIPE
Interoperability

Evaluation Criteria:

1) Test: [G1366.1]

Does the Web service user digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

2) Test: [G1366.2]

Does the Web service provider digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

G1367

Statement:

Digitally sign **message** fragments that are required not to change during transport.

Rationale:

Signing message fragments allows the consumer of the message fragment to verify the message fragment has not changed since the producer signed the message fragment.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
XML Web Service Security
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1367.1]

Do message fragments sent between producers and subscribers have digital signatures when the message content must remain unchanged during transport?

Procedure:

Check system requirments for message fragments that must be transmitted unchanged between the producer and consumer. For these message frangments, check that digital signature are used to detect changes to the message fragments.

Example:

None

G1369

Statement:

Digitally sign all requests made to a security token service.

Rationale:

Prevent hackers from intercepting a message and requesting a security token.

Referenced By:

Interoperability
Other Design Tenets
XML Web Service Security
Design Tenet: Identity Management, Authentication, and Privileges
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1369.1]

Does the Web service user digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

2) Test: [G1369.2]

Does the Web service provider digitally sign all messages?

Procedure:

Generate a test message and check it for digital signatures.

Example:

None

G1371

Statement:

Use the **Digital Signature Standard** for creating **Digital Signatures**.

Rationale:

Following Industry standards ensures interoperability.

Referenced By:

Design Tenet: Encryption and HAIPE
Interoperability
XML Web Service Security
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1371.1]

Does the Web service user generate signatures using the Digital Signature Standard?

Procedure:

Generate a test message and check it for compliance with the Digital Signature Standard.

Example:

None

2) Test: [G1371.2]

Does the Web service provider generate signatures using the Digital Signature Standard?

Procedure:

Generate a test message and check it for compliance with the Digital Signature Standard.

Example:

None

G1372

Statement:

Use an X.509 **Certificate** to pass a **Public Key**.

Rationale:

This ensures that the owner passing the key is who he says.

Referenced By:

XML Web Service Security
Other Design Tenets
Design Tenet: Identity Management, Authentication, and Privileges
Maintainability
Design Tenet: Encryption and HAIPE
Interoperability

Evaluation Criteria:

1) Test: [G1372.2]

Does the Web service provider send a public key as part of its messages?

Procedure:

Generate a test message and check it for an X.509.

Example:

None

2) Test: [G1372.1]

Does the Web service user send a public key as part of its messages?

Procedure:

Generate a test message and check it for an X.509.

Example:

None

G1373

Statement:

Encrypt messages that cross an **IA** boundary.

Rationale:

Prevent hackers from reading sensitive information.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
XML Web Service Security
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1373.1]

Does the Web service user encrypt all messages?

Procedure:

Generate a test message and check it for encryption.

Example:

None

2) Test: [G1373.2]

Does the Web service provider encrypt all messages?

Procedure:

Generate a test message and check it for encryption.

Example:

None

G1374

Statement:

Individually **encrypt** sensitive **message** fragments intended for different intermediaries.

Rationale:

Individually encrypting message fragments allows targeting individual fragments at different intermediaries along the message path to the final destination.

Referenced By:

Interoperability
XML Web Service Security
Design Tenet: Encryption and HAIPE
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1374.1]

Are sensitive fragments of the message encrypted?

Procedure:

Observe messages that are sent to see if the sensitive fragments of the message are encrypted.

Example:

None

G1376

Statement:

Do not **encrypt** key elements that are needed for correct **SOAP** processing.

Rationale:

It is possible to encrypt the entire SOAP message, various portions of the SOAP message or the contents of the data transported within the SOAP message. Encrypting the entire SOAP message requires that any intermediate processing of the SOAP message requires decryption of the entire message.

Referenced By:

XML Web Service Security
Design Tenet: Encryption and HAIPE
Interoperability
Other Design Tenets

Evaluation Criteria:

1) Test: [G1376.1]

Does the Web service user encrypt the entire message?

Procedure:

Generate a test message and check it to make sure the XML tags are not encrypted.

Example:

None

2) Test: [G1376.2]

Does the Web service provider encrypt the entire message?

Procedure:

Generate a test message and check it to make sure the XML tags are not encrypted.

Example:

None

G1377

Statement:

Use **LDAP** 3.0 or later to perform all connections to LDAP repositories.

Rationale:

Using industry-proven LDAP standards helpe ensure interoperability of the directory repository with its consumers. LDAP v3 addresses some of the limitations of LDAP v2 in the areas of internationalization and authentication. It also allows adding new features without also requiring changes to the existing protocol through the use of using extensions and controls while maintaining backward compatibility with LDAP v2.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Reusability
LDAP Security

Evaluation Criteria:

1) Test: [G1377.1]

Check port 636 if supporting secure LDAP (SLDAP)

Procedure:

Test the connection using an SLDAP client.

Example:

None

G1378

Statement:

Encrypt communication with **LDAP** repositories.

Rationale:

Encryption of communication to LDAP servers helps prevent disclosure of data during transmission.

Referenced By:

Maintainability
Interoperability
Design Tenet: Encryption and HAIPE
Design Tenet: Identity Management, Authentication, and Privileges
LDAP Security

Evaluation Criteria:

1) Test: [G1378.1]

Are connections to LDAP repositories encrypted?

Procedure:

Verify that connections to LDAP repository use Transport Layer Security (TLS) or Secure Sockets Layer (SSL).

Example:

G1379

Statement:

Use **SAML** version 2.0 for representing security assertions.

Rationale:

SAML 2.0 supports **XML** assertions for supporting cross domain access and Web services. The value of this type of access is that the passing of an assertion eliminates the need to create another account in another domain.

Referenced By:

Interoperability
Design Tenet: Mediate Security Assertions
Security Assertion Markup Language (SAML)
Design Tenet: Cross-Security-Domains Exchange

Evaluation Criteria:

1) Test: [G1379.1]

Can the SAML message be validated against SAML V2.0 schema?

Procedure:

Validate SAML message against SAML V2.0.

Example:

G1380

Statement:

Use the **XACML** 2.0 standard for **SAML**-based rule engines.

Rationale:

XACML-based rules can define the mechanism for creating the rule and policy set that enable meaningful **authorization** decisions. XAMCL is also integrated with **SAML** to support **role-based access control** or hierarchical resources, such as portions of XML documents.

Referenced By:

Design Tenet: Mediate Security Assertions
Interoperability
Security Assertion Markup Language (SAML)
Design Tenet: Identity Management, Authentication, and Privileges
Design Tenet: Cross-Security-Domains Exchange

Evaluation Criteria:

1) Test: [G1380.1]

Does the SAML-based rules engine use the XACML 2.0 standard?

Procedure:

Emulate a rule and run against rule engine using SOAP messaging.

Example:

G1381

Statement:

Encrypt all sensitive persistent data.

Rationale:

When data is persisted, there is always a chance that the security of the system that stores the data may be compromised. To minimize the risk, all sensitive data such as passwords and personal information should be encrypted when it is persisted.

Referenced By:

Interoperability
Design Tenet: Encryption and HAIPE
Data Tier

Evaluation Criteria:

1) Test: [G1381.1]

Is all sensitive data that is persisted encrypted?

Procedure:

Look at all data stores and check for encrypted passwords and other sensitive data..

Example:

G1382

Statement:

Be associated with one or more **Communities of Interest** (**COIs**).

Rationale:

The DoD Net-Centric Data Strategy emphasizes the establishment of Communities of Interest (**COI**s). This strategy introduces management of data within Communities of Interest (COIs) rather than standardizing **data elements** across the DoD. Thus all DoD Programs must map to one of more COIs. DoD Programs should participate in COIs as a normal course of doing business. They will identity relevant COIs; actively collaborate with them to promote reuse and cross-coordination of **metadata**; sponsor participation of system developers in the COI process and where appropriate contribute engineering expertise to the COI as a stakeholder. New programs should include community collaboration requirements in acquisition documents as required.

Referenced By:

Design Tenet: Make Data Interoperable
Design Tenet: Be Responsive to User Needs
Design Tenet: Make Data Understandable
Reusability
Metadata Registry
Interoperability

Evaluation Criteria:

1) Test: [G1382.1]

Is the Program associated with a **COI**?

Procedure:

Check the DoD Metadata registry to determine whether program is associated with any **COI**(s).

Example:

None

G1383

Statement:

Use a **registered namespace** in the XML Gallery in the **DoD Metadata Registry**.

Rationale:

The use of the **DoD Metadata Registry** helps to avoid name collisions and conflicts.

The assignation of a unique **registered namespace** permits a program to be uniquely identified and categorized. The DoD's Net-Centric Data Strategy requires that data products be stored in shared spaces to provide access to all authorized users and that these data products be tagged with **metadata** to enable discovery of data by authorized users. The use of a unique registered namespace provides an absolute identifier to products associated with a particular product and is an **XSD** schema requirement.

Referenced By:

Interoperability
Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Reusability
Metadata Registry
Design Tenet: Make Data Visible
Using XML Namespaces
Design Tenet: Make Data Accessible
Design Tenet: Make Data Trustable
Design Tenet: Provide Data Management
Design Tenet: Be Responsive to User Needs

Evaluation Criteria:

1) Test: [G1383.1]

Does the Program have an assigned namespace for its XML data assets?

Procedure:

Check **DoD Metadata Registry** to determine whether the Program is associated with **COI**(s).

Example:

None

G1384

Statement:

Review **XML Information Resources** in the **DoD Metadata Registry**, using those which can be reused.

Rationale:

The DoD Net-Centric Data Strategy requires that **XML** information resources within a **COI** in the **DoD Metadata Registry** be examined by DoD projects for possible reuse to help foster common standards within a **COI** and promote interoperability.

Note: *The proposed DoD Metadata Registry tools have not been formally released. The Beta version thereof is in testing. Automatic Waivers of this requirement will be permitted until the tools are formally released.*

Referenced By:

Design Tenet: Make Data Interoperable
Interoperability
Reusability
Design Tenet: Provide Data Management
Design Tenet: Make Data Understandable
Design Tenet: Be Responsive to User Needs
Metadata Registry
Using XML Namespaces

Evaluation Criteria:

1) Test: [G1384.1]

Has the program reused information resources from the **DoD Metadata Registry**?

Procedure:

Check the **XSDs** associated with the program to determine whether XSDs referenced by other namespaces have been used. Check the **DoD Metadata Registry** to determine whether the Program has registered the reuse of XML information resources belonging to other namespaces. Reuse is indicated by formally subscribing to selected components in the registry.

Example:

None

G1385

Statement:

Identify **XML Information Resources** for registration in the XML Gallery of the **DoD Metadata Registry**.

Rationale:

The DoD Net-Centric Data Strategy requires that **XML Information Resources** developed during the course of a program be identified, examined for usefulness by other DoD Programs in the same or related **COIs** and be submitted for inclusion in the XML Gallery of the **DoD Metadata Registry**.

Referenced By:

Design Tenet: Provide Data Management
Design Tenet: Make Data Interoperable
Metadata Registry
Design Tenet: Make Data Trustable
Interoperability
Design Tenet: Make Data Visible
Design Tenet: Make Data Accessible
Using XML Namespaces
Reusability

Evaluation Criteria:

1) Test: [G1385.1]

Has the Program submitted new information resources to the **DoD Metadata Registry**?

Procedure:

Check the **XSDs** associated with the program namespace to determine whether they have been registered in the **DoD Metadata Registry** XML Gallery.

Example:

None

G1386

Statement:

Review predefined commonly used **data elements** in the **Data Element Gallery** of the **DoD Metadata Registry**, using those in the **relational database** technology which can be reused in the Program.

Rationale:

The DoD Net-Centric Data Strategy requires that DoD Programs examine data element information resources within a **COI** in the **DoD Metadata Registry** for possible reuse to help foster common standards within a **COI** and promote interoperability. Elements include **`US State Codes`** and **`Country Codes`**. This reuse is preferential to reusing existing industry standard **data elements** or developing new **data elements**.

Referenced By:

Design Tenet: Provide Data Management
Design Tenet: Be Responsive to User Needs
Reusability
Design Tenet: Make Data Understandable
Interoperability
Metadata Registry
Design Tenet: Make Data Interoperable

Evaluation Criteria:

1) Test: [G1386.1]

Has the Program reused common database elements?

Procedure:

Check the DoD Metadata Registry Data Element Gallery to determine whether the program has registered database elements for reuse. Reuse is indicated by formally subscribing to selected components in the registry.

Check the program database to see whether registered have been included therein.

Example:

None

G1387

Statement:

Identify **data elements** created during Program development for registering in the **Data Element Gallery** of the **DoD MetaData Registry**.

Rationale:

The DoD Net-Centric Data Strategy requires that Programs identify and examine developed **data elements** for usefulness by other DoD Programs in the same or related **COIs** and submit the data elements for inclusion in the **Data Element Gallery** of the **DoD Metadata Registry**.

Referenced By:

Design Tenet: Make Data Visible
Interoperability
Metadata Registry
Design Tenet: Make Data Accessible
Design Tenet: Make Data Trustable
Design Tenet: Provide Data Management
Reusability

Evaluation Criteria:

1) Test: [G1387.1]

Has the Program submitted common database elements to the **DoD Metadata Registry**?

Procedure:

Check the DoD Metadata Registry Data Element Gallery to determine whether the program has submitted database elements for reuse.

Example:

None

G1388

Statement:

Use predefined commonly used database tables in the **DoD Metadata Registry**.

Rationale:

The DoD Net-Centric Data Strategy requires that DoD Programs examine data table information resources within a **COI** in the **DoD Metadata Registry** for possible reuse to help foster common standards within a COI and promote interoperability. This reuse is preferable to reusing existing industry standard **data elements** or developing new data elements. Some examples are **`Country Code`**, **`US State Code`**, **`Purchase Order Type Code`**, **`Security Classification Code`**. These tables are found in the **Reference Data Set** Gallery of the DoD Metadata Registry.

Referenced By:

Design Tenet: Make Data Understandable
Design Tenet: Be Responsive to User Needs
Metadata Registry
Reusability
Interoperability
Design Tenet: Make Data Trustable
Design Tenet: Make Data Interoperable

Evaluation Criteria:

1) Test: [G1388.1]

Has the Program reused common database tables?

Procedure:

Check the DoD Metadata Registry to determine whether the program has registered database tables for reuse. Reuse is indicated by formally subscribing to selected components in the registry.

Check the program database to see whether registered data tables have been included therein.

Example:

None

G1389

Statement:

Publish database tables which are of common interest by registering them in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.

Rationale:

The DoD Net-Centric Data Strategy requires that DoD Programs identify and examine developed data tables for usefulness by other DoD Programs in the same or related **COIs** and be submit the data elements for inclusion in the **Reference Data Set** Gallery of the **DoD Metadata Registry**.

Referenced By:

Design Tenet: Make Data Accessible
Design Tenet: Provide Data Management
Design Tenet: Make Data Visible
Design Tenet: Make Data Interoperable
Design Tenet: Make Data Trustable
Interoperability
Metadata Registry
Design Tenet: Make Data Understandable
Design Tenet: Be Responsive to User Needs
Reusability

Evaluation Criteria:

1) Test: [G1389.1]

Has the Program submitted common database tables to the DoD Metadata Registry?

Procedure:

Check the DoD Metadata Registry Reference Data Set Gallery to determine whether the program has submitted database tables for reuse.

Example:

None

G1390

Statement:

Standardize on the terminology published by relevant **Communities of Interest** (**COIs**) listed in the **Taxonomy Gallery** of the **DoD Metadata Registry**.

Rationale:

A **taxonomy** partitions the body of knowledge associated with a **Community of Interest COI** and defines the relationships among component parts. A taxonomy permits classification of concepts associated with a COI. This in turn provides categories and definitions for discovery tags which aids in information use and retrieval by authorized users. Program use of COI taxonomies occurs in several places:

1. Taxonomy used to describe information services for discovery.

2. Taxonomies created by the COI as a means to extend the **DoD Discovery Metadata Specification** (**DDMS**) for data asset discovery.

3. Taxonomies used to support mediation.

Referenced By:

Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Design Tenet: Provide Data Management
Metadata Registry
Design Tenet: Make Data Accessible
Design Tenet: Be Responsive to User Needs

Evaluation Criteria:

1) Test: [G1390.1]

Has the Program adhered to the standard **taxonomies** for the **COIs** associated with the program?

Procedure:

Check the DoD Metadata Registry and Taxonomy Gallery to determine whether taxonomies exist for the COI in which the Program resides.

Example:

None

G1391

Statement:

Identify **taxonomy** additions or changes in conjunction with the **Communities of Interest** (**COIs**) during the Program development for potential inclusion in the **Taxonomy Gallery** of the **DoD Metadata Registry**.

Rationale:

DoD Programs associated with a specific COI need to identify and submit potential taxonomy changes or additions to the **DoD Metadata Registry** to maintain an accurate and effective taxonomy within the **COI**.

Referenced By:

Design Tenet: Make Data Visible
Design Tenet: Make Data Accessible
Design Tenet: Be Responsive to User Needs
Design Tenet: Make Data Interoperable
Metadata Registry
Design Tenet: Make Data Understandable

Evaluation Criteria:

1) Test: [G1391.1]

Has the Program submitted **taxonomy** additions or changes to the **DoD Metadata Registry**?

Procedure:

Check the DoD Metadata Registry and to determine whether the program has submitted taxonomy changes for reuse.

Example:

None

G1566

Statement:

Use **`alt`** attributes to provide alternate text for non-text items such as images.

Rationale:

This usage aids users in understanding the Web page even if their browsers cannot display images.

Referenced By:

Human Factor Considerations for Web-Based User Interfaces
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1566.1]

Are alt attributes provided for non-text content?

Procedure:

Check for the existence of alt attributes for all Web site non-text content.

Example:

None.

G1569

Statement:

Maintain a comprehensive list of all of the **Components** that are part of the Node.

Rationale:

Throughout the lifecycle of a Node (from design to instantiation), this action is fundamental to the provisioning of a shared infrastructure and the avoidance of functional duplication within the Node. This activity has a direct impact on the design and implementation requirements during acquisition.

Referenced By:

Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Reusability
Nodes as Stakeholders
Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test: [G1569.1]

Is there a list of Components that comprise the Node?

Procedure:

Examine the documents (for example, the Node's design requirements) and look for a list of Components.

Example:

None.

G1570

Statement:

Assume an active management role among the **Components** within the Node.

Rationale:

Involvement of the Node as a stakeholder in its Components (from design to instantiation) has a bearing on **Global Information Grid** (GIG) interoperability. Strong coordination among a Node's Components will likely avoid the external exposure of inconsistencies or, worse, incomplete, inaccurate, or misunderstood data.

Referenced By:

Nodes as Stakeholders
Interoperability

Evaluation Criteria:

1) Test: [G1570.2]

Do the Components of the Node set forth requirements in their [appropriate acquisition document] for coordinating with the Node.

Procedure:

Check the [appropriate acquisition document] of the Components and determine if the Node is listed as a stakeholder or if there are requirements for coordinating with the Node.

Example:

A Component's **Capability Development Document** (**CDD**) may state a requirement for participating in a Node which could satisfy this requirement.

2) Test: [G1570.1]

Do the Components of the Node list the Node as a primary stakeholder in their [appropriate acquisition document]?

Procedure:

Check the [appropriate acquisition document] of the Components and determine if the Node is listed as a stakeholder or if there are requirements for coordinating with the Node.

Example:

A Component's **Capability Development Document** (**CDD**) may state a requirement for participating in a Node which could satisfy this requirement.

G1571

Statement:

Maintain a comprehensive list of all the **Communities of Interest** (COIs) to which the **Components** of a Node belong.

Rationale:

The Node infrastructure must be engineered to support the information exchange between **Communities of Interests** (COIs). If a comprehensive list of COIs is not created and maintained then the infrastructure may no longer be adequate and may continue to make provisions for COIs that are no longer a part of the Node.

Referenced By:

Net-Centric Information Engineering
Design Tenet: Be Responsive to User Needs

Evaluation Criteria:

1) Test: [G1571.1]

Do the Node's Components have representation registered within the DoD Metadata Registry as members of the Communities of Interest (COIs)?

Procedure:

Examine the DoD Metadata Registry for members of the Node organization that are members of the pertinent COIs.

Example:

None.

G1572

Statement:

Include the Node as a party to any **Service Level Agreements** (SLAs) signed by any of the **Components** of the Node.

Rationale:

The Node has a stake in performance specifications provided in the **Service Level Agreements** (SLA). Since the SLA is a contract that commits the application service provider to a required level of service. The Node must be able to support that level of service with its infrastructure.

Referenced By:

Net-Centric Information Engineering
Design Tenet: Scalability
Design Tenet: Availability

Evaluation Criteria:

1) Test: [G1572.1]

Does the Node have copies of all Service Level Agreements (SLAs) signed by its Components?

Procedure:

Compare the Service Level Agreements (SLAs) against the service Components supported by the Node.

Example:

None.

G1573

Statement:

Define the enterprise design patterns that a Node supports.

Rationale:

The Node infrastructure must be engineered to support information exchanges between various **Communities of Interest** (COIs). The COIs can require any number of **Components** to fulfill the COIs mission, When a Component wishes to make its data available over the **enterprise**, there are different enterprise design pattern which can be used. For example, the mechanism selected by a Component to exchange information may be publish-subscribe, broker, or client server. The Node infrastructure must support whichever enterprise design pattern mechanism is selected.

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Net-Centric Information Engineering

Evaluation Criteria:

1) Test: [G1573.1]

Does the Node document which types of enterprise design patterns it supports?

Procedure:

Look through the Node documents for a list of enterprise design patterns it supports.

Example:

None.

G1574

Statement:

Define which enterprise design patterns a **Component** requires.

Rationale:

A Component should document which enterprise design patterns it intends to capitalize on to meet its mission. For example, a client interested in using a client-server weather service, could have problems if the weather service is a real-time publish-subscribe service. This action clarifies for the Node which enterprise design patterns are required by its Components and provides direction for which patterns to support at the Node level.

Referenced By:

Net-Centric Information Engineering
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1574.1]

Does the Component indicate which type of enterprise design pattern it will use?

Procedure:

Look through the Component documentation and that defines what type of enterprise design pattern it uses.

Example:

None.

G1575

Statement:

Designate Node representatives to relevant **Communities of Interest** (COIs) in which Components of the Node participate.

Rationale:

COI is the inclusive term used to describe collaborative groups of users who must exchange information in pursuit of their shared goals, interests, missions, or business processes and who therefore must have shared vocabulary for the information they exchange. The principal mechanism for recording COI agreements is the **DoD Metadata Registry** required by the DoD CIO Memorandum *DoD Net-Centric Data Management Strategy: Metadata Registration*. There are registry implementations on the Non-secure Internet Protocol Router Network (**NIPRNet**), Secret Internet Protocol Router Network (**SIPRNet**), and Joint Worldwide Intelligence Communications System (**JWICS**).

Referenced By:

Net-Centric Information Engineering
Design Tenet: Be Responsive to User Needs

Evaluation Criteria:

1) Test: [G1575.1]

Does the Node have representation registered within the Metadata Registry as members of the **Communities of Interest** (COIs)?

Procedure:

Examine the **DoD Metadata Registry** for members of the Node organization that are members of the pertinent COIs.

Example:

None.

G1576

Statement:

Provide an environment to support the development, build, integration, and test of net-centric capabilities.

Rationale:

Nodes should provide an environment to support the development, integration, and testing of net-centric capabilities of its **Components**. As Nodes themselves and the Components within the Nodes move closer to the implementation of net-centric capabilities, it becomes increasingly important to provide a development, integration, and test environment to support those capabilities. This environment should allow for the exercise not just the Node infrastructure, but also either host locally within the Node, or provide access to, **Net-Centric Enterprise Services** (NCES) piloted services. The particulars on how this is done depend on the characteristics of the Node. For example, mobile or deployed Nodes would provide environments substantially different than fixed land-based or permanent Nodes.

Referenced By:

Internal Component Environment
CES Definitions and Status
Maintainability
Design Tenet: Joint Net-Centric Capabilities

Evaluation Criteria:

1) Test: [G1576.1]

Are there instructions on how to develop, build, integrate or test Components within the Node?

Procedure:

Look for user guides or installation instructions that cover the Node environment.

Example:

None.

G1577

Statement:

Maintain an **Enterprise Service** schedule for interim and final **enterprise** capabilities within the Node.

Rationale:

The current state of **Enterprise Services** is in flux. Developing **Components** that rely on those services can create a circular problem for development. An enterprise service schedule for interim and final capabilities will help elevate the co-dependencies of the Component lifecycle from the Node lifecycle.

Referenced By:

Maintainability
Coordination of Node and Enterprise Services
Internal Component Environment
CES Parallel Development

Evaluation Criteria:

1) Test: [G1577.1]

Is there an enterprise service schedule or roadmap that covers interim and final capabilities of the Node?

Procedure:

Look for the existence of the schedule or a roadmap for the Node.

Example:

None.

G1578

Statement:

Define a schedule for **Components** that includes the use of the **Enterprise Services** defined within the Node's enterprise service schedule.

Rationale:

The exercise of matching those **Enterprise Services** required by the **Component** to those provided by the Node can help identify and gaps in the Node's functionality. By tying the Component's enterprise services to the Node's **enterprise** schedule, critical paths may be identified in the Node's schedule.

Referenced By:

CES Parallel Development
Coordination of Node and Enterprise Services
Maintainability
Internal Component Environment

Evaluation Criteria:

1) Test: [G1578.1]

Does the Component have an enterprise service schedule or roadmap that shows the progression of enterprise service usage by interim and final capabilities of the Component?

Procedure:

Look for the existence of the schedule or a roadmap for the Component.

Example:

None.

G1579

Statement:

Define which **Enterprise Services** the Node will host locally when the Node becomes operational.

Rationale:

Locally defined **Enterprise Services** are inherently faster and less susceptible to network failures and traffic than local services. If a **Component** requires performance based or critical enterprise services that the Node will only provide as a **proxy**, then development, building, integration and testing should be done to the local enterprise service specification. If the Node developed enterprise service will not be ready until near the end of the Component's schedule, take steps to minimize risk.

Referenced By:

Internal Component Environment
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1579.1]

Does the Node specification identify which Enterprise Services will be locally defined within the Node?

Procedure:

Review the Node specification for a list of Enterprise Services that will be locally defined within the Node.

Example:

None.

G1580

Statement:

Define which **Enterprise Services** will be hosted over the **Global Information Grid** (GIG) when the Node becomes operational.

Rationale:

Enterprise Services that are defined using **proxies** should have interfaces that follow the standards defined by the enterprise service provider. Therefore, the access to the **server** should be fairly stable and almost static in nature with few changes. These are services that should be in the critical path of a Component's mission.

Referenced By:

Internal Component Environment
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [G1580.1]

Does the Node specification identify which Enterprise Services will be defined using proxies?

Procedure:

Review the Node specification for a list of Enterprise Services that will be defined using proxies.

Example:

None.

G1581

Statement:

Expose **legacy system** or **application** functionality through the use of a service that uses a **facade design pattern**.

Rationale:

Nodes might contain **systems** or **applications** that are in the **Sustainment** lifecycle phase. These **Components** are often referred to as ***legacy*** systems or applications. If a Node needs to expose functionality or data form the legacy Component, changing the internals of such Components to support net-centricity is often impractical with little return on investment. This design pattern offers a reasonable interim solution.

Referenced By:

Integration of Legacy Systems
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability

Evaluation Criteria:

1) Test: [G1581.1]

Does the Node use **facade design patterns** such as the wrapper or adapter pattern to expose the functionality of legacy systems or applications?

Procedure:

Make sure that all the Components that are exposed to the internal Node Components or to the external network (with the Node as a proxy) use a facade design pattern such as wrapper or adapter.

Example:

None.

G1582

Statement:

In Node **Enterprise Service** schedules, include version numbers of standard Enterprise Services interfaces being implemented.

Rationale:

Given the complexity, varied implementation timing, and leading edge nature of **Enterprise Services**, the **orchestration** of efforts is essential for the successful integration of the Node's Components. The dependencies captured by such a schedule should clearly show what capabilities will be available and when during the Node's lifecycle.

Referenced By:

Design Tenet: Network Connectivity
Maintainability
Coordination of Node and Enterprise Services

Evaluation Criteria:

1) Test: [G1582.1]

Are Enterprise Services interface versions provided on the enterprise service schedule for the Node?

Procedure:

Review the Enterprise Services schedule published for the Node and make sure the schedule provides necessary details including specific version numbers, workarounds, assumptions, constraints and configuration limitations that are interwoven into the schedule.

Example:

An Enterprise Service might be releasing a new version during the lifecycle of the Node's development; which version's functionality will be available when is essential for the successful integration of the Node's Components.

2) Test: [G1582.2]

Are Enterprise Services interface versions provided on the enterprise service schedule for the Component?

Procedure:

Review the Enterprise Services schedule published for the Component and make sure the schedule provides necessary details including specific version numbers, workarounds, assumptions, constraints and configuration limitations that are interwoven into the schedule.

Example:

An Enterprise Service might be releasing a new version during the lifecycle of the Node's development; which version's functionality will be available when is essential so the Component can utilize the appropriate available capabilities.

G1583

Statement:

Provide routine **Enterprise Services** schedule updates to every **Component** of a Node.

Rationale:

A fundamental justification for the existence of nodes is to ensure it provides a shared infrastructure for its Components. If that infrastructure evolves independently of the Components, then they may be developed at timeframes and rates of evolution that differ from the capabilities of the available shared infrastructure. In addition, Components may be members of multiple Nodes, providing an additional coordination challenge. Regular updates to the Componetns of the master schedule will assist in managing this challenge.

Referenced By:

Maintainability
Coordination of Internal Components

Evaluation Criteria:

1) Test: [G1583.1]

Are there multiple iterations of the Enterprise Services schedule developed over time and is the most recent update timely?

Procedure:

Check for version numbering and release dates of the Enterprise Services schedule. Ensure that a reasonably recent update is available.

Example:

None.

G1584

Statement:

Provide a transport infrastructure that is shared among **Components** within the Node.

Rationale:

Transport elements provided by the Node are a means for the Node to implement **Global Information Grid** (GIG) **Information Assurance** (IA) boundary protections, bind Components together, and satisfy other enterprise requirements. As transport elements are an essential piece of the net-centric puzzle, they also play a key role in minimizing interoperability issues. A Node's provisioning of the shared transport and related guidance is a key aspect of its existence.

Referenced By:

Design Tenet: Transport Goal
Node Transport

Evaluation Criteria:

1) Test: [G1584.1]

Does the Node's design provide for a transport infrastructure?

Procedure:

Review the Node's infrastructure design and ensure that the Node provides the necessary transport elements for shared use by its Components.

Example:

None.

2) Test: [G1584.2]

Are the Node's Components using the Node provisioned transport infrastructure?

Procedure:

Review the design of the Node's Components (see G1569) and ensure that they all utilize the common transport infrastructure of inter-Nodal communication.

Example:

None.

G1585

Statement:

Provide a transport infrastructure for the Node that implements **Global Information Grid** (GIG) **Information Assurance** (IA) boundary protections.

Rationale:

The **Global Information Grid** (GIG) is intended to be the ***outside world*** for all the Components within the Node. In order to protect the Components within the Node from the outside world and to protect the outside world from the Node, the Node should control the **IA** Boundary.

Referenced By:

Node Transport
Design Tenet: Net-Centric IA Posture and Continuity of Operations
Design Tenet: Transport Goal

Evaluation Criteria:

1) Test: [G1585.1]

Is there an IA device in the acquisition list?

Procedure:

L

ook for an IA device within the parts list for the Node.

Example:

None.

2) Test: [G1585.2]

Is the IA device configured to meet security requirements?

Procedure:

Check the Node's IA installation guide and look for procedures that describe how to configure the IA device for the Nodes particular needs.

Example:

None.

G1586

Statement:

Provide a transport infrastructure for the Node that is **Internet Protocol Version 6** (IPv6) capable in accordance with the appropriate governing transition plan.

Rationale:

During the transition period in the DoD community (FY06-FY15) networks, services and applications will be in a mixed environment. All Critical **Key Performance Parameters** (KPPs) must be able to operate in an **Internet Protocol Version 4** (IPv4) only network, an **Internet Protocol Version 6** (IPv6) only network, and a dual-stack network.

Referenced By:

Design Tenet: IPv6
Design Tenet: Transport Goal
IPv4 to IPv6 Transition

Evaluation Criteria:

1) Test: [G1586.1]

Does the system operate in an Internet Protocol Version 6 (IPv6) only Network?

Procedure:

Critical Functions will be tested in a Network that only supports Internet Protocol Version 6 (IPv6). The host must be able to complete all critical functions utilizing only IPv6 on the network (no tunneling).

Example:

None.

G1587

Statement:

Prepare an **Internet Protocol Version 6** (IPv6) transition plan for the Node.

Rationale:

The transition from **Internet Protocol Version 4** (IPv4) to **Internet Protocol Version 6** (IPv6) is non-trivial and requires a great deal of coordination and effort on the part of everyone involved. The transition plan helps to minimize the potential disastrous side effects of the transition.

Referenced By:

IPv4 to IPv6 Transition
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1587.1]

Is there an Internet Protocol Version 6 (IPv6) transition plan for the Node?

Procedure:

Look for an Internet Protocol Version 6 (IPv6) transition plan document.

Example:

None.

G1588

Statement:

Coordinate an **Internet Protocol Version 6** (IPv6) transition plan for a Node with the **Components** that comprise the Node.

Rationale:

The effects of the transition from **Internet Protocol Version 4** (IPv4) to **Internet Protocol Version 6** (IPv6) is isolated in the Node infrastructure but can have impacts on all the **Components** that comprise the Node. The transition Plan should cover a "window" that allows all the Components to operate in either IPv4 or IPv6 (i.e., **Dual Stack Mode**) to make the transition.

Referenced By:

IPv4 to IPv6 Transition
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1588.1]

Does the plan allow for a **Dual Stack** environment at least during some transition period?

Procedure:

Look for a part of the transition plan that addresses **Dual Stack** mode of operation.

Example:

None.

G1589

Statement:

Address issues in the appropriate governing **IPv6** transition plan as part of the Internet Protocol Version 6 (IPv6) Transition Plan for a Node.

Rationale:

DoD has mandated that each service create an **IPv6** transformation office to manage the transition to IPv6. Node transition plans must be aligned and in conformance with the appropriate governing office's plans or criteria.

Referenced By:

IPv4 to IPv6 Transition
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1589.1]

Does the Node's IPv6 Transition Plan have a section that addresses specific criteria established by the appropriate governing IPv6 transition office or plan?

Procedure:

Review the IPv6 plan for a section or specific criteria that address the appropriate items from the appropriate governing plan or is approved by the appropriate governing office.

Example:

The Air Force IPv6 Transition Office requires each program to develop a plan with approval by the transition office (in lieu of aligning with a central plan). To check an Air Force Node's alignment, look to see that the Node's IPv6 transition plan is approved by the appropriate authority.

G1590

Statement:

Include transition of all the impacted elements of the network as part of the **Internet Protocol Version 6** (IPv6) Transition Plan for a Node.

Rationale:

Internet Protocol Version 6 (IPv6) transition has an impact on many transport infrastructure **Components**. The Node's IPv6 Transition Plan should include transition of all impacted network elements including **DNS**, routing, security, and dynamic address assignment. The DoD IPv6 Network Engineer's Guidebook (Draft) and the DoD IPv6 Application Engineer's Guidebook (Draft) provide guidance for transition of impacted Components.

Referenced By:

IPv4 to IPv6 Transition
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1590.1]

Does the Internet Protocol Version 6 (IPv6) Transition Plan address the impact of the transition to IPv6 on the Domain Name Service (DNS)?

Procedure:

Review the plan and look for a section dedicated to the Domain Name Service (DNS). At a minimum, it should indicate that there is no impact.

Example:

None.

2) Test: [G1590.2]

Does the Internet Protocol Version 6 (IPv6) Transition Plan address the impact of the transition to IPv6 on routing?

Procedure:

Review the plan and look for a section dedicated to routing. At a minimum, it should indicate that there is no impact.

Example:

None.

3) Test: [G1590.3]

Does the Internet Protocol Version 6 (IPv6) Transition Plan address the impact of the transition to IPv6 on security?

Procedure:

Review the plan and look for a section dedicated to security. At a minimum, it should indicate that there is no impact.

Example:

None.

4) Test: [G1590.4]

Does the Internet Protocol Version 6 (IPv6) Transition Plan address the impact of the transition to IPv6 on dynamic address assignment?

Procedure:

Review the plan and look for a section dedicated to dynamic address assignment. At a minimum, it should indicate that there is no impact.

Example:

None.

G1591

Statement:

Prepare **IPv6** Working Group products as part of the Internet Protocol Version 6 (IPv6) transition plan for a Node.

Rationale:

The **Internet Protocol Version 6** (IPv6) Working Group has prescribed various products that can aid in the planning for the transition from **Internet Protocol Version 4** (IPv4) to IPv6. The Node's Transition Plan should prepare these products to ensure that all the required activities are addressed.

Referenced By:

IPv4 to IPv6 Transition
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1591.1]

Are the Internet Protocol Version 6 (IPv6) Working Group products in the Node's Transition Plan?

Procedure:

Look for the Working Group products in the Node's Transition Plan.

Example:

None.

G1592

Statement:

Include interoperability testing in the plan as part of the **Internet Protocol Version 6** (IPv6) transition plan for a Node.

Rationale:

During the **DoD** transition period, a mixed **IPv4**/IPv6 environment will exist. Interoperability testing with both standards will ensure the Node can fully function during the transition period with all other Nodes.

Referenced By:

Design Tenet: IPv6
IPv4 to IPv6 Transition

Evaluation Criteria:

1) Test: [G1592.1]

Does the Node's IPv6 transition plan address interoperability testing in a mixed environment?

Procedure:

Review the transition plan and verify that a test plan exists that specifically addresses interoperability testing in a mixed IP environment.

Example:

None.

G1595

Statement:

Implement **Domain Name System** (DNS) to manage hostname/address resolution within the Node.

Rationale:

Using **Domain Name System** (DNS) obviates the need for hard-coding **Internet Protocol** (IP) addresses within the Node. In addition, DNS servers local to the Node allow for stable access of replicated entries from outside the Node.

Referenced By:

Domain Name System (DNS)
Design Tenet: Packet Switched Infrastructure
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1595.2]

Are there any hard coded Internet Protocol (IP) addresses within the source code or data files?

Procedure:

Look at the source code, properties files and descriptor files for the occurrence of Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) Internet Protocol (IP) addresses.

Example:

None.

2) Test: [G1595.1]

Is there a Domain Name System (DNS) server in the Node acquisition list?

Procedure:

Look for a Domain Name System (DNS) server within the parts list for the Node.

Example:

None.

G1596

Statement:

Use **Domain Name System** (DNS) **Mail eXchange (MX) Record** capabilities to configure electronic mail delivery to the Node.

Rationale:

Utilizing the **Domain Name System** (**DNS**) **Mail eXchange (MX) record** capability will avoid the need to hard code delivery routes and instructions within a Node's email system and buffers it from physical changes made to email delivery points and routes outside of the Node. The DNS MX record is a standard and commonly accepted mechanism for resolving email delivery routes and addresses across the Internet.
Internet Engineering Task Force (**IETF**) Request fo Comments (RFC) 2821 of April 2001 established rules for MX record usage.

Referenced By:

Domain Name System (DNS)
Design Tenet: Packet Switched Infrastructure

Evaluation Criteria:

1) Test: [G1596.1]

Are there **Mail eXchange (MX) Records** defined within the **Domain Name System** (DNS)?

Procedure:

Look at the Domain Name System (DNS) records for Mail eXchange (MX) Records.

Example:

None.

G1598

Statement:

Allow dynamic **Domain Name System** (DNS) updates to the Node's internal DNS service by local **Dynamic Host Configuration Protocol** (DHCP) **server(s)**.

Rationale:

There are two basic methods for assigning of **Internet Protocol** (IP) addresses within a network: static and dynamic. Static addresses are assigned to a particular system and never change. Dynamic Internet Protocol (IP) addresses are issued for a variable length of time: the ***DCHP lease time***. **Dynamic Host Configuration Protocol** (DHCP) is the principle mechanism used to assign and manage dynamic IP addresses. If the DHCP servers are allowed to update the **Domain Name System** (DNS), then the number of static addresses required by the system can be drastically reduced with preference being given to requesting services by domain name rather than IP address.

Referenced By:

Design Tenet: Packet Switched Infrastructure
Domain Name System (DNS)

Evaluation Criteria:

1) Test: [G1598.1]

Does the Domain Name System (DNS) server in the Node acquisition list support updates from Dynamic Host Configuration Protocol (DHCP) Servers?

Procedure:

Review the Domain Name System (DNS) server specification to confirm that it supports such operations.

Example:

None.

G1599

Statement:

Support both **Internet Protocol Version 4** (IPv4) and **Internet Protocol Version 6** (IPv6) simultaneously in the Node's **Domain Name System** (DNS) service.

Rationale:

During the transition period in the DoD community (FY06-FY15) networks, services and applications will be in a mixed environment. The Domain Name System (DNS) returns different address records depending on the Internet Protocol (IP) environment: A records for IPv4 or AAAA records for IPv6. A DNS must be able to support both.

Referenced By:

IPv4 to IPv6 Transition
Domain Name System (DNS)
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1599.1]

Does the Domain Name System (DNS) server support both A and AAAA records?

Procedure:

Review the Domain Name System (DNS) specification to confirm that it supports both A and AAAA records.

Example:

None.

G1600

Statement:

Obtain from DISA any and all **Internet Protocol Version 6** (IPv6) addresses used on DoD systems in the Node.

Rationale:

All the **Internet Protocol** (IP) addresses in use on a DoD network must be from an appropriate clearing house in order to maintain control and accountability on the network. **DISA** is the clearing house for all DoD addresses.

Referenced By:

IPv4 to IPv6 Transition
Domain Name System (DNS)
Design Tenet: IPv6

Evaluation Criteria:

1) Test: [G1600.1]

Is there a proper entry in the Military Network Information Center (MILNIC) for every IP address assigned to the system?

Procedure:

Verify an adequate address allocation has been made inMilitary Network Information Center (MILNIC) for the system.

Example:

None.

G1601

Statement:

Use configurable **routers** to provide dynamic **Internet Protocol** (IP) address management using **Dynamic Host Configuration Protocol** (DHCP).

Rationale:

There are two basic methods for assigning of **Internet Protocol** (IP) addresses within a network: static and dynamic. Static addresses are assigned to a particular system and never change. Dynamic IP addresses are issued for a variable length of time: the ***DCHP lease time***. **Dynamic Host Configuration Protocol** (DHCP) is the principle mechanism used to assign and manage dynamic IP addresses.

Referenced By:

Design Tenet: Inter-Network Connectivity
Routers
Design Tenet: Network Connectivity
Multicast
Design Tenet: Packet Switched Infrastructure

Evaluation Criteria:

1) Test: [G1601.1]

Does the router in the Node acquisition list support Dynamic Host Configuration Protocol (DHCP)?

Procedure:

Review the router specification to confirm that it supports such operations.

Example:

None.

G1602

Statement:

Use configurable **routers** to provide static **Internet Protocol** (IP) addresses.

Rationale:

Some network **Components** such as the **routers** themselves and other security related services must reside on static **Internet Protocol** (IP) addresses. Serious comprises in the network can arise if these services are allowed to be dynamic.

Referenced By:

Design Tenet: Inter-Network Connectivity
Routers
Design Tenet: Network Connectivity
Design Tenet: Packet Switched Infrastructure

Evaluation Criteria:

1) Test: [G1602.1]

Does the **router** in the Node acquisition list support static **Internet Protocol** (IP) addressing?

Procedure:

Review the router specification to confirm that it supports such operations.

Example:

None.

G1604

Statement:

Use configurable **routers** to provide time synchronization services using **Network Time Protocol** (NTP).

Rationale:

Over time, most computer clocks drift. **Network Time Protocol** (NTP) is one way to ensure that a computer clock stays accurate. Unfortunately, in order to stay synchronized, a network connection needs to be maintained. In environments that have limited bandwidth or poor **quality of service** (QoS) this can become a major issue.

Referenced By:

Time Services
Design Tenet: Inter-Network Connectivity
Design Tenet: Packet Switched Infrastructure
Routers
Design Tenet: Network Connectivity

Evaluation Criteria:

1) Test: [G1604.1]

Does the **router** in the Node acquisition list support NTP Service?

Procedure:

Review the routers specification to confirm that it supports such operations.

Example:

None.

G1605

Statement:

Use configurable **routers** to provide **multicast** addressing.

Rationale:

Multicast addresses identify interfaces that allow a packet to be sent to all the addresses registered for the multicast service. This allows network to easily support applications such as **collaboration**, audio and video.

Referenced By:

Routers
Design Tenet: Network Connectivity
Design Tenet: Packet Switched Infrastructure
Design Tenet: Inter-Network Connectivity

Evaluation Criteria:

1) Test: [G1605.1]

Does the **router** in the Node acquisition list support NTP Service?

Procedure:

Review the router specification to confirm that it supports such operations.

Example:

None.

G1606

Statement:

Manage **routers** remotely from within the **Node**.

Rationale:

Router manufactures routinely provide tools to enable remote configuration and management of the router. These tools are can speed and centralize the administration of the Nodes routers.

Referenced By:

Design Tenet: Inter-Network Connectivity
Routers
Design Tenet: Network Connectivity
Design Tenet: Decentralized Operations and Management
Design Tenet: Packet Switched Infrastructure

Evaluation Criteria:

1) Test: [G1606.1]

Does the **router** in the Node acquisition list support remote management?

Procedure:

Review the router specification to confirm that it supports such operations.

Example:

None.

G1607

Statement:

Configure routers according to **National Security Agency** (NSA) Router Security Configuration guidance.

Rationale:

The *Router Security Configuration Guide* provides technical guidance intended to help network administrators and security officers improve the security of their networks. It contains principles and guidance for secure configuration of **Internet Protocol** (IP) routers, with detailed instructions for Cisco System routers. The information presented can be used to control access, help resist attacks, shield other network **Components**, and help protect the integrity and confidentiality of network traffic.

Referenced By:

Design Tenet: Packet Switched Infrastructure
Design Tenet: Inter-Network Connectivity
Design Tenet: Concurrent Transport of Information Flows
Routers
Design Tenet: Network Connectivity
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1607.1]

Is the **Router** Security Checklist complete and up to date?

Procedure:

Check for the occurrence of the checklist; there should be a copy for every time the checklist has been completed. The checklist should indicate the date, time and results of the checklist with recommendation actions.

Example:

Router Security Checklist
This security checklist is designed to help review router security configuration and remind a user of any security areas that might be missed.

- Router security policy written, approved, distributed.
- Router IOS version checked and up to date.
- Router configuration kept off-line, backed up, access to it limited.
- Router configuration is well-documented, commented.
- Router users and passwords configured and maintained.
- Password encryption in use, enable secret in use.
- Enable secret difficult to guess, knowledge of it strictly limited. (if not, change the enable secret immediately)
- Access restrictions imposed on Console, Aux, VTYs.

- Unneeded network servers and facilities disabled.
- Necessary network services configured correctly (e.g. DNS)
- Unused interfaces and VTYs shut down or disabled.
- Risky interface services disabled.
- Port and protocol needs of the network identified and checked.
- Access lists limit traffic to identified ports and protocols.
- Access lists block reserved and inappropriate addresses.
- Static routes configured where necessary.
- Routing protocols configured to use integrity mechanisms.
- Logging enabled and log recipient hosts identified and configured.
- Router's time of day set accurately, maintained with NTP.
- Logging set to include consistent time information.
- Logs checked, reviewed, archived in accordance with local policy.
- SNMP disabled or enabled with good community strings and ACLs.

G1608

Statement:

Obtain the reference time for the Node time service from a globally synchronized time source.

Rationale:

Currently Network Time Service is not a ubiquitous service across the **Global Information Grid** (GIG). Security directives prevent IP-based time synchronization across **firewall** boundaries (e.g., AFI 33-115, 16). An example of a precise globally synchronized time source is a **Global Positioning System** (GPS) system.

Referenced By:

Design Tenet: Packet Switched Infrastructure
Time Services
Design Tenet: Inter-Network Connectivity
Design Tenet: Network Connectivity

Evaluation Criteria:

1) Test: [G1608.1]

Does the Node acquisition list include a precise globally synchronized time source such as **Global Positioning System** (GPS) system?

Procedure:

Review the acquisition list for a precise globally synchronized time source such as a **Global Positioning System** (GPS) system that can be used to accurately synchronize time.

Example:

None.

G1609

Statement:

Arrange for a backup time source for the Node time service.

Rationale:

The most common type of backup time sources are crystal oscillators. The physical characteristics of the piezoelectric quartz crystal produce electrical oscillations at an extremely accurate frequency. This frequency can be used to mark time.

Referenced By:

Design Tenet: Network Connectivity
Design Tenet: Packet Switched Infrastructure
Design Tenet: Inter-Network Connectivity
Time Services

Evaluation Criteria:

1) Test: [G1609.1]

Does the Node acquisition list include a backup time system?

Procedure:

Review the acquisition list for a backup time system that can be used to synchronize time accurately. For example: crystal oscillator, cesium or rubidium crystal oscillators. Crystal oscillator types and their abbreviations:

MCXO	microcomputer-compensated crystal oscillator
OCVCXO	oven-controlled voltage-controlled crystal oscillator
OCXO	oven-controlled crystal oscillator
RbXO	rubidium crystal oscillators (RbXO)
TCVCXO	temperature-compensated-voltage controlled crystal oscillator
TCXO	temperature-compensated crystal oscillator
VCXO	voltage-controlled crystal oscillator

Example:

None.

G1610

Statement:

Configure the **Dynamic Host Configuration Protocol** (DHCP) services to assign **multicast** addresses.

Rationale:

When **Dynamic Host Configuration Protocol** (DHCP) services assign temporary **Internet Protocol** (IP) addresses to clients, the clients may wish to participate in a **multicast** service. Therefore, the DHCP service must support the assignment of multicast addresses as part of normal operations.

Referenced By:

Design Tenet: Network Connectivity
Multicast
Design Tenet: Packet Switched Infrastructure
Design Tenet: Inter-Network Connectivity

Evaluation Criteria:

1) Test: [G1610.1]

Does the **router** in the Node acquisition list support the assignment of **multicast** Internet Protocol (**IP**) addresses as part of the normal **Dynamic Host Configuration Protocol** (DHCP) service?

Procedure:

Review the **router** specification to confirm that it supports such operations.

Example:

None.

G1611

Statement:

Implement Internet Protocol (**IP**) gateways to interoperate with the **Global Information Grid** (GIG) until IP is supported natively for **Components** that are not IP networked, such as aircraft data links (**Link-16**, **SADL**, etc.).

Rationale:

Component systems such as aircraft data links (Link-16, SADL, etc), should implement **Transmission Control Protocol/Internet Protocol** (TCP/IP) gateways to interoperate with the **Global Information Grid** (GIG) until TCP/IP is supported natively. This acts as an interim step that can be used to bridge the **Internet Protocol** (IP) divide.

Referenced By:

Design Tenet: Packet Switched Infrastructure
Integration of Non-IP Transports

Evaluation Criteria:

1) Test: [G1611.1]

Is there an Internet Protocol (IP) gateway in the system?

Procedure:

Identify **Transmission Control Protocol/Internet Protocol** (TCP/IP), **User Datagram Protocol** (UDP) or **DDS** code that will be front-ended by a gateway.

Example:

None.

G1612

Statement:

Implement Internet Protocol (**IP**) gateways as a **service**.

Rationale:

This does not mean that the service is a **Web service** or that it is limited to request/reply or other such usage patterns. In fact, for high-frequency data, such as track reporting, a function of the service could be to set up an out-of-band communication with a subscriber.

Referenced By:

Integration of Non-IP Transports
Design Tenet: Packet Switched Infrastructure

Evaluation Criteria:

1) Test: [G1612.1]

Is the gateway developed as a service that could be advertised in a registry?

Procedure:

Examine the gateway and determine if it is a service.

Example:

None.

G1613

Statement:

Prepare a **Node** to host new **Component** services developed by other Nodes or by the **enterprise** itself.

Rationale:

A key aspect of an open systems approach to interoperability is **modular design** which is also a basic tenet of good development practice. Modularity will support the dynamic redeployment of a **Component** into different Nodes that requires the capabilities of the Component thus promoting broader interoperability between different Nodes and Components. Where possible, Nodes should adopt standards based, platform independent frameworks that facilitate ***pluggable*** deployment capabilities for Components so it can leverage the capabilities developed elsewhere.

Referenced By:

Cross-Domain Interoperation
Design Tenet: Cross-Security-Domains Exchange
Web Client Platform

Evaluation Criteria:

1) Test: [G1613.1]

Does the Node support the elements of a modern component based framework such as **Java Platform, Enterprise Edition** (Java EE), **.NET** or **CORBA**?

Procedure:

Look for the existence of Java Platform, Enterprise Edition (Java EE), .NET or CORBA frameworks with in the Node's Component list or in its delivered software.

Example:

None.

G1619

Statement:

Configure **clients** with a **Common Access Card** (**CAC**) reader.

Rationale:

DoD Instruction 8520.2, *Public Key Infrastructure (PKI) and Public Key (PK) Enabling* [R1206] , defines **Common Access Card** (CAC) applicability and scope, in part, as follows:

This Instruction applies to:... 2.4. All DoD unclassified and classified information systems including networks (e.g., Non-secure Internet Protocol (IP) Router Network , Secret Internet Protocol Router Network, Web servers, and e-mail systems. Excluded are Sensitive Compartmented Information, and information systems operated within the Department of Defense that fall under the authority of the Director of Central Intelligence Directive (DCID) 6/3 (reference (h)).

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Common Access Card (CAC) Reader

Evaluation Criteria:

1) Test: [G1619.1]

Do all the **client** and **server** hardware come equipped with **Common Access Card** (CAC) Readers?

Procedure:

Review the hardware list and verify that all hardware comes with or has external CAC readers.

Example:

None.

G1622

Statement:

Implement **commercial off-the-shelf** (COTS) software that protects against malicious code on each operating system in the Node in accordance with the Desktop Application **Security Technical Implementation Guide** (STIG).

Rationale:

The viral and worm assault on computing resources is major concern but is not strictly limited to DoD hardware and operating systems. It has become a ubiquitous, wide spread problem that spreads destruction indiscriminately. Since the problem is not strictly a DoD problem, **commercial off-the-shelf** (COTS) solutions are always being updated to meet the current threats and are essential in protecting the assets. All hardware platforms should employ virus and worm detection and removal software that is routinely run (especially on hardware the runs Microsoft products).

Note: *For purposes of this guidance, anti virus software includes related update and maintenance capabilities typically available with such packages.*

Referenced By:

Host Information Assurance
Other Design Tenets

Evaluation Criteria:

1) Test: [G1622.1]

Do all hardware devices listed in the Node acquisition list have COTS licensed virus and worm detection software?

Procedure:

Review the Node acquisition list and make sure there is one license for each piece of computer hardware.

Example:

None.

2) Test: [G1622.2]

Do all hardware devices listed in the Node acquisition list have COTS virus and worm detection software installed?

Procedure:

Review the prerequisites in the installation manual for virus and worm software.

Example:

None.

G1623

Statement:

Implement personal **firewall** software on **client** or **server** hardware used for remote connectivity in accordance with the Desktop Applications, Network and Enclave **Security Technical Implementation Guides** (**STIGs**).

Rationale:

All hardware that is plugged into a network is subject to attack by hackers. In addition to hardware **firewalls**, every piece of hardware should be protected by a software firewall. These firewalls continuously monitor the activity on the network port and detect possible hostile attacks. The user has the discretion to block hostile attacks permanently or for a particular occasion. Since this problem is not restricted to DoD assets, **Commercial off-the-shelf** (**COTS**) products are continuously being updated to meet the latest threats and are essential in meeting these threats.

.

Referenced By:

Host Information Assurance
Other Design Tenets
Design Tenet: Decentralized Operations and Management
Design Tenet: Inter-Network Connectivity

Evaluation Criteria:

1) Test: [G1623.1]

Do all the hardware devices listed in the Node acquisition list have COTS software firewall licensed software?

Procedure:

Review the Node acquisition list and make sure there is one license for each piece of computer hardware.

Example:

None.

2) Test: [G1623.2]

Do all hardware devises listed in the Node acquisition list have COTS **firewall** software installed and is it enabled?

Procedure:

Review the prerequisites in the installation manual for firewall software.

Example:

None.

G1624

Statement:

Install anti-**spyware** on all **client** and **server** hardware.

Rationale:

Spyware is a category of malicious software that can impact a system's operation in ways similar to virus and other intrusions. Extending the principles of protection against viruses and other intrusions to spyware is an essential activity to ensure stable system operation and security.

Referenced By:

Other Design Tenets
Host Information Assurance

Evaluation Criteria:

1) Test: [G1624.1]

Do all the hardware devices listed in the Node acquisition list have COTS software anti-spyware licensed software?

Procedure:

Review the Node acquisition list and make sure there is one license for each piece of computer hardware.

Example:

None.

2) Test: [G1624.2]

Do all hardware devices listed in the Node acquisition list have COTS anti-spyware software installed and is it enabled?

Procedure:

Review the prerequisites in the installation manual for firewall software.

Example:

None.

G1626

Statement:

Identify which **Core Enterprise Services** (CES) capabilities the Node **Components** require.

Rationale:

A Node needs to determine the set of **Core Enterprise Services** (CES) its **Components** will require in order to ensure efficient prioritization of activities and resources to provide those services. NCES has defined a set of common capabilities that help categorize types of services that may be required by a Node's Components. Identification of the capabilties required by Components will help the Node determine which Services will need to be implemented.

Referenced By:

Design Tenet: Open Architecture
CES Definitions and Status

Evaluation Criteria:

1) Test: [G1626.1]

Does the list of Components that comprise the Node indicate which CES capabilities are required to deploy each Component?

Procedure:

Review the list of Components and verify that they have indicated which CES capabilities are required to support the Component.

Example:

None.

G1627

Statement:

Identify the priority of each **Core Enterprise Services** (CES) capability the Node **Components** require.

Rationale:

Identifying the priority of capabilities required by the Node's **Components** will assist the Node in allocation of scarce resources towards the delivery of CES in the Node and minimize risks during deployment of Components within the Node. Some capabilities are ***essential*** at getting a Component Deployed at a Node. Some are essential for a particular Component increment. With this information the Node can construct a schedule that supports the transition and evolution of the current federation of systems to the **Global Information Grid** (GIG) vision.

Referenced By:

Design Tenet: Open Architecture
CES Parallel Development
CES Definitions and Status

Evaluation Criteria:

1) Test: [G1627.1]

Does the list of Components that comprise the Node indicate the priority of the CES capabilities either relative to each other or as of a date?

Procedure:

Review the list of Components and verify that they have indicated what the priority of the CES capabilities either relative to each other or as of a date.

Example:

None.

G1629

Statement:

Identify which **Net-Centric Enterprise Services** (NCES) capabilities the Node requires during deployment.

Rationale:

Relying on a high-bandwidth **Transmission Control Protocol/Internet Protocol** (TCP/IP) network connection is not a reality for many deployed Nodes. These Nodes will have to develop many of their own CES capabilities for use by their member **Components** while deployed. When the Node is not deployed, it may rely on proxies to the **Net-Centric Enterprise Services** (NCES) services.

Referenced By:

CES Definitions and Status
Design Tenet: Joint Net-Centric Capabilities
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1629.1]

Does the Node have a list of **Net-Centric Enterprise Services** (NCES) capabilities that it depends on while deployed?

Procedure:

Review the Node's documents for a list of Net-Centric Enterprise Services (NCES)capabilities required by the Node while deployed.

Example:

None.

G1630

Statement:

Comply with the applicable **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) for implemented **Core Enterprise Services** (CES) in the Node.

Rationale:

When a **CES** is implemented locally, use the **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) developed by **DISA** as the authoritative definition of the interfaces. This allows a **Component** that is hosted by one Node to be hosted on another Node with a minimal impact.

Referenced By:

Design Tenet: Open Architecture
CES and Intermittent Availability
Key Interface Profile (KIP)

Evaluation Criteria:

1) Test: [G1630.1]

Do all **CES** used locally within the Node implement the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for Core Enterprise Services (CES) implement Global Information Grid (GIG) Key Interface Profiles (KIPs) for that CES.

Example:

None.

G1631

Statement:

Expose **Core Enterprise Services** (CES) that comply with the applicable **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in all Node services **proxies**.

Rationale:

A Node may expose or control access to **Global Information Grid** (GIG) **CES** by using **proxies**. This allows a **Component** that is hosted by one Node to be hosted on another Node with a minimal impact.

Referenced By:

Design Tenet: Open Architecture
Key Interface Profile (KIP)
CES and Intermittent Availability

Evaluation Criteria:

1) Test: [G1631.1]

Do all **CES proxies** locally defined within the Node expose CES using the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for CES proxies follow Key Interface Profiles (KIPs) for that Global Information Grid (GIG) KIP.

Example:

None.

G1632

Statement:

Certify and accredit Nodes with all applicable DoD **Information Assurance** (IA) processes.

Rationale:

Nodes are part of the DoD **Global Information Grid** (GIG) and are consequently required to have DoD **Information Assurance** (IA) certification and accreditation. Details for certification and accreditation are specified in DoD Directive 8500.1, DoD Instruction 8500.2, DoD Directive 8580.1, and DoD Instruction 5200.40. Satisfaction of these requirements results in IA compliance verification of the Node.

Referenced By:

Other Design Tenets
Information Assurance (IA)
Design Tenet: Net-Centric IA Posture and Continuity of Operations

Evaluation Criteria:

1) Test: [G1632.1]

Does the Node have DoD **Information Assurance** (IA) certification and accreditation?

Procedure:

Ask to examine the certification and accreditation reports.

Example:

None.

G1633

Statement:

Host only DoD **Information Assurance** (IA) certified and accredited **Components**.

Rationale:

Nodes that expose the external Node users to non-certified or non-accredited **Components** represent a risk to the stability of the entire Node network and can introduce interoperability issues between Nodes (and related Components).

Referenced By:

Information Assurance (IA)
Other Design Tenets
Design Tenet: Net-Centric IA Posture and Continuity of Operations

Evaluation Criteria:

1) Test: [G1633.1]

Does the Node have a plan to scan all Components on a routine basis?

Procedure:

Look for a plan and examine the results of the scan.

Example:

None.

G1634

Statement:

Certify and accredit **Components** with all applicable DoD **Information Assurance** (IA) processes.

Rationale:

Each **Component** could theoretically be deployed on any Node. Therefore, it is the responsibility of the Component to be DoD **Information Assurance** (IA) certified and accredited.

Referenced By:

Information Assurance (IA)
Design Tenet: Net-Centric IA Posture and Continuity of Operations
Other Design Tenets

Evaluation Criteria:

1) Test: [G1634.1]

Are all the **Components** DoD **Information Assurance** (IA) certified and accredited?

Procedure:

Examine the certification and accreditation reports.

Example:

None.

G1635

Statement:

Make Nodes that will be part of the **Global Information Grid** (GIG) consistent with the *GIG Integrated Architecture*.

Rationale:

The **Global Information Grid** (GIG) architecture describes the basic, high level architecture in which Nodes reside. It is an integrated architecture consisting of the various **DoDAF** views. It provides a common lexicon and defines a basic infrastructure for the performance of information exchanges with other GIG Nodes using the **GIG Enterprise Services** (GES) and the **Net-Centric Enterprise Services** (NCES). The GIG Integrated Architecture is available via the DoD Architecture Repository System (DARS), https://dars1.army.mil/ [user account and PKI certificate required for access].

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Interoperability
Integrated Architectures

Evaluation Criteria:

1) Test: [G1635.1]

Are there **DoDAF** integrated architecture products defined for the Node that are consistent with the **GIG** Integrated Architecture?

Procedure:

Look for the occurrence of **Operational View** (OV), Systems **and ServicesView** (SV), **Technical Standards View** (TV) and **All Views** (AV).

Example:

None.

G1636

Statement:

Comply with the **Net-Centric Operations and Warfare Reference Model** (NCOW RM).

Rationale:

The **Net-Centric Operations and Warfare Reference Model** (NCOW RM) is focused on achieving net-centricity. Compliance with the NCOW RM translates to articulating how each Node approaches and implements net-centric features. Compliance does not require separate documentation; rather, it requires that a Node address, within existing architecture, analysis, and program architecture documentation, the issues identified by using the model, and further, make explicit the path to net-centricity the program is taking.
Node compliance with the NCOW RM is demonstrated through inspection and analysis:

- Use of NCOW RM definitions and vocabulary;
- Incorporation of NCOW RM Operational View (OV) capabilities and services in the materiel solution;
- Incorporation of NCOW RM Technical View **Information Technology** (IT) and **National Security Systems** (NSS) standards in the Technical View products developed for the materiel solution.

Compliance with the NCOW RM is a critical component of compliance with the **Net-Ready Key Performance Parameter** (NR-KPP).

Referenced By:

Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Net-Centric Operations and Warfare Reference Model (NCOW RM)

Evaluation Criteria:

1) Test: [G1636.2]

Have the instructions in Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 3170.01 been used to check the Node for Net-Centric Operations and Warfare Reference Model (NCOW RM) compliance?

Procedure:

Check Node documentation.

Example:

2) Test: [G1636.3]

Have the instructions in Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 6212.01 been used to check the Node for Net-Centric Operations and Warfare Reference Model (NCOW RM) compliance?

Procedure:

Check Node documentation.

Example:

3) Test: [G1636.1]

Have the instructions in the Defense Acquisition University (DAU) Guidebook section 7.2.6 been used to check the Node for NCOW RM compliance?

Procedure:

Check Node documentation.

Example:

G1637

Statement:

Make Node-implemented **directory services** comply with the directory services **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs).

Rationale:

When directory services are implemented locally, use the **Global Information Grid** (GIG) **KIPs** developed by DISA as the authoritative definition of the interfaces. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Directory Services
Interoperability

Evaluation Criteria:

1) Test: [G1637.1]

Do all directory services used locally within the Node implement the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for directory services implement Global Information Grid (GIG) Key Interface Profiles (KIPs) for that directory services.

Example:

None.

G1638

Statement:

Comply with the directory services **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node directory services **proxies**.

Rationale:

A Node may expose or control access to **Global Information Grid** (GIG) directory services by using **proxies**. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Directory Services
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability

Evaluation Criteria:

1) Test: [G1638.1]

Do all directory services **proxies** locally defined within the Node expose directory services using the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for directory services proxies follow Key Interface Profiles (KIPs) for that Global Information Grid (GIG) KIPs.

Example:

None.

G1639

Statement:

Describe **Components** exposed by the Node as specified by the **Service Definition Framework**

Rationale:

The construction of registry entries is specified by the **Service Definition Framework** (SDF) documented in Net-Centric Implementation Directives (NCIDs) S300. The common Service Definition Framework that serves as the basis for adequately describing the offered **Component** service from both a provider's and consumer's perspective. It describes the contract between the Component service provider and the Component service consumer, and serves as the basis for a **Service Level Agreement** (SLA). The common service definition framework consists of elements that include interface, service level, security and implementation information.

Referenced By:

Service Discovery
Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test: [G1639.1]

Is there a **Service Definition Framework** (SDF) available for each of the Components' Services exposed through the Node?

Procedure:

Look for a Service Definition Framework (SDF) for each Component service exposed through the Node.

Example:

None

G1640

Statement:

Register **Components** exposed by the Node with the **DISA**-hosted registries.

Rationale:

The best way to for an exposed Node's **Component** service to be discovered is by being registered in the DISA registry. The DISA registry implementation uses **Universal Description, Discovery, Integration** (UDDI).

Referenced By:

Interoperability
Service Discovery

Evaluation Criteria:

1) Test: [G1640.1]

Is the exposed Node's Component's service registered in the DISA **Universal Description, Discovery, Integration** (UDDI) Registry?

Procedure:

Examine the DISA Universal Description, Discovery, Integration (UDDI) Registry and look for the exposed Node's Component's service.

Example:

None.

G1641

Statement:

Comply with the Service Discovery **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node-implemented **Service Discovery** (SD).

Rationale:

When a **Service Discovery** (SD) is implemented locally, the **Global Information Grid** (GIG) Kips developed by DISA should be used as the authoritative definition of the interfaces. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Service Discovery
Design Tenet: Service-Oriented Architecture (SOA)
Interoperability

Evaluation Criteria:

1) Test: [G1641.1]

Does the **Service Discovery** (SD) used locally within the Node implement the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for Service Discovery (SD) implement Global Information Grid (GIG) Key Interface Profiles (KIPs) for that Service Discovery.

Example:

None.

G1642

Statement:

Comply with the **Service Discovery Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node Service Discovery (SD) **proxies**.

Rationale:

A Node may expose or control access to **Global Information Grid** (GIG) **Service Discovery** (SD) by using **proxies**. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Design Tenet: Service-Oriented Architecture (SOA)
Service Discovery
Interoperability

Evaluation Criteria:

1) Test: [G1642.1]

Do the **Service Discovery** (SD) **proxies** locally defined within the Node expose Service Discovery using the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for Service Discovery (SD) proxies follow KIPs for that Global Information Grid (GIG) Key Interface Profiles (KIPs).

Example:

None.

G1643

Statement:

Comply with the **Federated Search** # **Registration Web Service** (RWS) **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node implemented Federated Search # Registration Web Service (RWS).

Rationale:

When a **Federated Search** # **Registration Web Service** (RWS) is implemented locally, use the **Global Information Grid** (GIG) KIPs developed by **DISA** as the authoritative definition of the interfaces. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Content Discovery Services

Evaluation Criteria:

1) Test: [G1643.1]

Does a **Federated Search** # **Registration Web Service** (RWS) used locally within the Node implement the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for Federated Search # Registration Web Service (RWS) implement Global Information Grid (GIG) Key Interface Profiles (KIPs) for that Federated Search # Registration Web Service (RWS).

Example:

None.

G1644

Statement:

Comply with the **Federated Search** # **Search Web Service** (SWS) **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) in Node implemented Federated Search # Search Web Service (SWS).

Rationale:

When a **Federated Search** # **Search Web Service** (SWS) is implemented locally, use the **Global Information Grid** (GIG) **Key Interface Profiles** (KIPs) developed by **DISA** as the authoritative definition of the interfaces. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Interoperability
Content Discovery Services

Evaluation Criteria:

1) Test: [G1644.1]

Does **Federated Search** # **Search Web Service** (SWS) used locally within the Node implement the applicable **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Verify that the interfaces for Federated Search # Search Web Service (SWS) implement Global Information Grid (GIG) Key Interface Profiles (KIPs) for that Federated Search # Search Web Service (SWS).

Example:

None.

G1645

Statement:

Implement a local **Content Discovery Service** (CDS).

Rationale:

The node should implement the **Content Discovery Service** (CDS) as part of the node infrastructure to be shared among the **Components** hosted at the Node. A CDS will allow other Nodes and Components to find content within the node. The systems within the Node normally provide the content.

> ***Note:*** *If a Node is frequently disconnected, has intermittent connectivity, or is otherwise isolated, then hosting a local CDS might not be a practical solution for external content discovery and more effective means for internal discovery may be applicable.*

Referenced By:

Interoperability
Content Discovery Services

Evaluation Criteria:

1) Test: [G1645.1]

Does the Node implement the **Content Discovery Service** (CDS) **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Look for an implementation at the Node of the Content Discovery Service (CDS) Global Information Grid (GIG) Key Interface Profiles (KIPs).

Example:

None.

G1646

Statement:

Comply with the directory services **Global Information Grid** (**GIG**) **Key Interface Profiles (KIPs**) in Node **Federated Search** Services **proxies**.

Rationale:

A Node may expose or control access to **Global Information Grid** (**GIG**) **Federated Search** Services by using **proxies**. This allows a **Component** that is hosted by one Node to be hosted on another node with a minimal impact.

Referenced By:

Interoperability
Content Discovery Services

Evaluation Criteria:

1) Test: [G1646.1]

Do all **Federated Search** Services **proxies** locally defined within the Node expose Federated Search Services using the applicable **Global Information Grid KIP**?

Procedure:

Verify that the interfaces for Federated Search Services proxies follow KIPs for that Global Information Grid (GIG) Key Interface Profiles (KIPs).

Example:

None.

G1647

Statement:

Provide access to the **Federated Search** Services.

Rationale:

Content Discovery Service can search across a set of Content Discovery Services and yield an integrated result. The current approach to providing this service is to harness an existing capability termed ***Federated Search*** developed under the **Horizontal Fusion** (HF) program. The capability utilizes the **DoD Discovery Metadata Specification** (DDMS).

Referenced By:

Content Discovery Services
Design Tenet: Provide Data Management

Evaluation Criteria:

1) Test: [G1647.1]

Does the Node provide access to the **Federated Search** Service **Global Information Grid** (GIG) **Key Interface Profile** (KIP)?

Procedure:

Look for a proxy or an implementation that provides access to the ***Federated Search***

Example:

None.

G1652

Statement:

Use DoD **PKI** X.509 **certificates** for **servers**.

Rationale:

Using a DoD PKI X.509 **server certificate** identifies the server as being trusted by the DoD and guarantees that the server's identity is legitimate.

Referenced By:

Identity Management
Design Tenet: Identity Management, Authentication, and Privileges

Evaluation Criteria:

1) Test: [G1652.1]

Is the server certificate a valid DoD PKI X.509 certificate that is non-expired?

Procedure:

Open the server certificate and check that it is trusted by a trusted DoD root certificate.

Example:

G1662

Statement:

Follow the guidance provided in the **Security Technical Implementation Guide** (STIG) for **Domain Name System** (DNS) implementations.

Rationale:

As a fundamental common service on **IP**-based networks, **DNS** is often a focal point for network attackers. Following the **STIG** ensures alignment with DoD identified security practices and configurations. The STIG addresses implementation options such as the choice of basic DNS server types (primary, secondary, caching-only), use of a split-DNS design, location of servers in the network and relationship to other network components, secure administration, security of zone transfers, and initial configuration.

Referenced By:

Domain Name System (DNS)
Other Design Tenets

Evaluation Criteria:

1) Test: [G1662.1]

Do the Node's **DNS** services follow the **STIG** for DNS implementations?

Procedure:

Compare Node DNS services configuration with those recommended by the STIG.

Example:

None.

G1667

Statement:

Implement **Virtual Private Networks** (VPNs) in accordance with the guidance provided in the Network **Security Technical Implementation Guide** (STIG).

Rationale:

Virtual Private Networks provide a means for Node access to users outside the security enclave. To Network **STIG** provides recommendations on how to configure VPNs for secure access.

Referenced By:

Virtual Private Networks (VPN)
Other Design Tenets

Evaluation Criteria:

1) Test: [G1667.1]

Does the configuration of the Node's **VPN** servers follow the recommendations of the Network **STIG**?

Procedure:

Check VPN server configuration against recommended configurations in the Network STIG.

Example:

None.

G1713

Statement:

Use an **Operating Environment** (**OE**) for all SCA applications that includes middleware that, at a minimum, provides the services and capabilities specified by Minimum CORBA Specification version 1.0.

Rationale:

Using a CORBA provider that adheres to the minimum CORBA v1.0, specification improves the interoperability between SCA Operating Environments.

Referenced By:

Software Communication Architecture
Design Tenet: RF Acquisition
Interoperability
Design Tenet: Service-Oriented Architecture (SOA)
Design Tenet: Open Architecture
Composeability
Reusability
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [G1713.1]

Does the OE contain middleware that provides the services and capabilities of minimum CORBA?

Procedure:

Check for minimum CORBA compliance in the CORBA provider's documentation.

Example:

G1714

Statement:

Develop **Software Communications Architecture** (**SCA**) applications to use only **Operating Environment** functionality defined by the SCA Application Environment Profile.

Rationale:

The SCA Application Environment Profile (AEP) is a subset of the Portable Operating System Interface (POSIX) specification. Functionality that is not part of the AEP is not guaranteed to be part of the operating environment. Applications that rely on functionality that is not part of the AEP will require changes to deploy or port to other SCA platforms.

Referenced By:

Software Communication Architecture
Design Tenet: Open Architecture
Reusability
Design Tenet: Service-Oriented Architecture (SOA)
Composeability
Design Tenet: Accommodate Heterogeneity
Design Tenet: RF Acquisition
Interoperability

Evaluation Criteria:

1) Test: [G1714.1]

Does the SCA application use Operating Environment functions not defined by a Application Environment Profile?

Procedure:

Check to see that all Operating Environment calls in the SCA application are listed in an Application Environment Profile.

Example:

G1717

Statement:

Use constants instead of hard-coded numbers for characteristics that may change throughout the lifetime of the model.

Rationale:

Constants increase the usefulness and lifetime of a design because the model can adapt to a variety of environments by postponing or modifying those parameters late in the design cycle. This makes the code more readable, maintainable and reusable.

Note: *This practice has been adapted from Cohen, section 1.6.1.1.3.*

Referenced By:

VHDL Coding and Design
Maintainability
Reusability

Evaluation Criteria:

1) Test: [G1717.1]

Are there any characteristics that are susceptible to modification that are directly given a value?

Procedure:

Parse the code and look for hard-coded characteristics that are susceptible to change and consider replacing them with a constant.

Example:

None

G1718

Statement:

Design circuits to be synchronous.

Rationale:

The preferred method of engineering today's digital ICs is based on a synchronous design. The main advantages of this are simplicity and reliability. Creating synchronous pieces of code increases interoperability and reusability when they are used with other synchronous modules.

Referenced By:

VHDL Synchronous Design
Maintainability
Reusability

Evaluation Criteria:

1) Test: [G1718.1]

Are all flip-flops clocked by the same, common clock signal?

Procedure:

Check to make sure a single external clock signal triggers the design to go from a well defined and stable state to the next one. On the active edge of the clock, all input and output signals and all internal nodes are stable in either the high or low state. Between two consecutive edges of the clock, the signals and nodes are allowed to change and may take any intermediate state.

Example:

None

G1719

Statement:

Automate testbench error checking in VHDL development.

Rationale:

Manual verification is subject to human error and is time consuming. In addition, automation promotes increased maintainability, because it enables fast and reliable verification of a model when modifications are made.

Note: *This practice has been adapted from Cohen, section 11.1.1.*

Referenced By:

VHDL Testbench
Composeability
Maintainability
Reusability

Evaluation Criteria:

1) Test: [G1719.1]

Does the testbench automatically report success or failure for each sub-test that it runs through?

Procedure:

Run the testbench to see if it automatically reports successes or failures for each sub-test.

Example:

None

G1724

Statement:

Develop XML documents to be well formed.

Rationale:

By W3C definition, XML documents must be well formed. However, documents that contain XML tags that are not well formed has no name and is often still referred to as an XML Document in common vernacular. Therefore, this guidance statements helps to clarify the need for well-formed documents. Well formed XML documents are those documents which have a proper XML syntax. This is essential if the XML is to be parsed using common, readily available open source and commercial XML parsers.

Referenced By:

Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Interoperability
XML Syntax
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1724.1]

Can the XML Document be parsed using a common, readily available XML Parser?

Procedure:

Open the XML document in a browser such as Mozilla Firefox or Microsoft Internet Explorer or use the XML Validator available from the W3 Schools at: http://www.w3schools.com/xml/xml_validator.asp

Example:

None

G1725

Statement:

Develop XML documents to be **valid** XML.

Rationale:

The content of a **valid** XML document conforms to a specific set of user-defined content rules contained in XML schemas. XML schemas describe data values correctness using predefined datatypes as base types and assigning values to the datatype specific attributes of those datatypes. For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and instead contains: alphanumeric text such as "hello"; is empty; or has other elements in its content, then the document is considered not valid.

Referenced By:

Design Tenet: Make Data Understandable
Defining XML Schemas
XML Instance Documents
XML Validation
Interoperability
Design Tenet: Open Architecture
Design Tenet: Make Data Interoperable

Evaluation Criteria:

1) Test: [G1725.1]

Does the document validation tool indicate that the XML document is valid?

Procedure:

Use a validating parser and verify that the document is valid.

Example:

None.

G1726

Statement:

Define XML Schemas using **XML Schema Definition** (XSD).

Rationale:

While it is possible to use **Document Type Definitions** (DTD) to convey much of the same information as the **XML Schema Definition** (XSD), XSDs have a several distinct advantages which are very useful in terms of interoperability. For example, DTDs do not capture domain or type range information very well (i.e. elevation in meters is from 0 to 12,000).

XML Schemas are a tremendous advancement over DTDs. Here are some of the reasons to use XSDs versus DTDs as delineated by Roger Costello in an XML tutorial (see the ***XML Schema Tutorial*** available at http://www.xfront.com):

- Enhanced datatypes support:
    - 44+ in XSDs versus 10 in DTDs
    - Support for user defined datatypes. For example, a user can define a new type based on the string type. Elements declared of this type must follow this specific pattern ddd-dddd, where d represents a numeric digit.
- Written using the same syntax as other XML instance documents. This means there is less to remember and more consistency with the same rules applying to all XML instance documents.
  XSDs support a limited Object-oriented (OO) paradigm. For example, new types can be derived from previously defined types with more or more stringent restrictions.
- Supports a kind of polymorphism where elements can be interchanged with parent or child elements. For example, a "Book" element can be substituted for the "Publication" element.
- Supports the definition of elements that are unordered collections or sets of other elements.
- Support for the identification of elements as part of a unique key.
- Support for elements that have the same name but different content
- Support for elements that have a null (i.e., nil) value.

Referenced By:

Design Tenet: Provide Data Management
Defining XML Schemas
Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Interoperability
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1726.1]

Are XML schemas defined using XML Schema Definitions?

Procedure:

Verify that XML schemas are defined using W3C XML Schema Definitions rather than Document Type Definitions.

Example:

None.

G1727

Statement:

Provide names for XML type definitions.

Rationale:

By naming type definitions in a schema, the type definitions can be reused in any number of other definitions. For example:

```
<xsd:complexType name="PointOfContact">
 <xsd:sequence>
   <xsd:element name="LastName" type="xsd:string"/>
   <xsd:element name="FirstName" type="xsd:string"/>
   <xsd:element name="MiddleName" type="xsd:string"/>
   <xsd:element name="NickName" type="xsd:string"/>
   <xsd:element name="PhoneNumber" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>
```

Can be reused anywhere a Point-Of-Contact needs to used. For Example:

```
<xsd:complexType name="Project">
 <xsd:sequence>
   <xsd:element name="ProjectName" type="xsd:string"/>
   <xsd:element name="ProgramManager" type="PointOfContact"/>
   <xsd:element name="HardwareManager" type="PointOfContact"/>
   <xsd:element name="SoftwareManager" type="PointOfContact"/>
   <xsd:element name="ConfigurationManager" type="PointOfContact"/>
 </xsd:sequence>
</xsd:complexType>
```

Referenced By:

Maintainability
Defining XML Types
Interoperability
Versioning XML Schemas
Design Tenet: Make Data Understandable
Design Tenet: Open Architecture

Evaluation Criteria:

1) Test: [G1727.1]

Do all complexTypes have names associated with them?

Procedure:

Examine all the complexType elements in the schema and verify that they have a name associated with them.

Example:

```
<xsd:complexType name="PointOfContact">
  ...
```

```
</xsd:complexType>
```

2) Test: [G1727.2]

Do all simpleTypes have names associated with them?

Procedure:

Examine all the simpleType elements in the schema and verify that they have a name associated with them.

Example:

```
<xsd:simpleType name="PointOfContact">
   ...
</xsd:simpleType>
```

G1728

Statement:

Define types for all **XML elements**.

Rationale:

There are two ways to associate the type-like information within an XML Schema. The first way is define an **XML element** as a global element of the schema element and the second is to define a complex or simple type. The first method violates G1727 and it does not support the clean separation of the definition of types from the use of the types.

By separating the definition of the types from the definition of the elements within structures, the types can be reused and are loosely coupled from any particular instance of the domain. The definitions of the type information can be maintained by a community that wishes to share the definition rather than any particular implementation or instance.

Referenced By:

Design Tenet: Make Data Understandable
Design Tenet: Open Architecture
Maintainability
Defining XML Types
Interoperability

Evaluation Criteria:

1) Test: [G1728.1]

Does the schema define any elements that are defined using references to other elements that are not part of a substitutionGroup rather than types?

Procedure:

Look for the use of an element's ref attribute.

Example:

None.

G1729

Statement:

Annotate XML type definitions.

Rationale:

Types in a schema represent a particular concept or aspect within a particular subject domain. Providing documentation about the type within the schema itself helps prevent disconnects between the documentation and the implementation as captured by the type definition.

Referenced By:

Design Tenet: Make Data Interoperable
Design Tenet: Make Data Understandable
Design Tenet: Provide Data Management
Design Tenet: Open Architecture
Maintainability
Defining XML Types

Evaluation Criteria:

1) Test: [G1729.1]

Do all the types defined within a schema have annotation that describes the nuances of type?

Procedure:

Look for an annotation for each simple type and complex type defined in the schema.

Example:

The complex type warranty includes an annotation that describes the purpose of the type and any caveats on when/how to use it.

G1730

Statement:

Follow an XML coding standard for defining schemas.

Rationale:

There are any number of coding standards that are defined for coding XML Schemas. Here are some areas covered by the most popular:

- Elements and Types are Upper Camel Case (UCC) convention.
- Type names end with the word Type.
- Attributes start with a lowercase letter and then revert to Lower Camel Case (LCC) convention.

Referenced By:

Maintainability
Defining XML Schemas
Interoperability

Evaluation Criteria:

1) Test: [G1730.1]

Is there a consistent XML coding convention followed when schemas are defined?

Procedure:

Look for the occurrence of a XML coding standard and verify that the XML Schemas follow the standard.

Example:

None.

G1731

Statement:

Only reference **XML elements** defined by a Type in substitution groups.

Rationale:

The 35mm, disk, and 3x5 components are simply declared as standalone **XML elements** which may be substituted for the abstract RecordingMedium element.

Note: *All of these RecordingMedium components have a type that is the same as, or derived from, the RecordingMediumType.*

Note: *The abstract RecordingMedium is associated with a type, RecordingMediumType, rather than defining the structure as part of the RecordingMedium element. This allows the definition of the RecordingMedium structure (i.e. type) to evolve independently.*

Referenced By:

Using XML Substitution Groups
Maintainability

Evaluation Criteria:

1) Test: [G1731.1]

Do substitutionGroup references point to an abstract element that has a structures defined by a type?

Procedure:

Ensure that all substitutionGroups point to an abstract element that has a structures defined by a type.

Example:

None.

G1735

Statement:

Use the **`.xsd`** file extension for files that contain XML Schema definitions.

Rationale:

It is possible to use any name for a schema file extension. However, using any extension other than **`.xsd`** causes confusion for humans as well as tools and utilities which rely on MIMEs often mapped to file extensions.

Referenced By:

Maintainability
XML Schema Files

Evaluation Criteria:

1) Test: [G1735.1]

Is the file extension that contains the schema definition .xsd?

Procedure:

Make sure that all XML documents that contain the xml **`schema`** tag have a file extension of .xsd.

Example:

None.

G1736

Statement:

Separate document schema definition and document instance into separate documents.

Rationale:

Separating the definition of the schema from the document instance supports the modularity by separating the definition of structure from the actual data. Each is allowed to evolve and change independently. In most cases, the definition of the structure of the data should be relatively static compared with the number of documents that are shared using that schema.

Document name: Camera.xsd

```
<xsd:schema
    targetNamespace="http://www.camera.org"
    elementFormDefault="qualified">
 <xsd:include schemaLocation="Nikon.xsd"/>
 <xsd:include schemaLocation="Olympus.xsd"/>
 <xsd:include schemaLocation="Pentax.xsd"/>
 <xsd:element name="Camera">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element
          name="Body"
          type="BodyType"/>
       <xsd:element
          name="Lens"
          type="LensType"/>
       <xsd:element
          name="ManualAdapter"
          type="ManualAdapterType"/>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Document name: Camera.xml

```
<?xml version="1.0"?>
<Camera xmlns ="http://www.camera.org"

        xsi:schemaLocation=
                   "http://www.camera.org
                    Camera.xsd">
 <Body>
   <Description>
      Ergonomically designed casing for easy handling
   </ Description>
 </Body>
 <Lens>
   <Zoom>300mm</Zoom>
   <F-Stop>1.2</F-Stop>
 </Lens>
 <ManualAdapter>
   <speed>1/10,000 sec to 100 sec</speed>
 </ManualAdapter>
</Camera>
```

Referenced By:

XML Schema Files

Maintainability
XML Instance Documents

Evaluation Criteria:

1) Test: [G1736.1]

Does the instance document have a <schema> tag?

Procedure:

Check the instance document and look for the use of the schema tag or the use of the XMLSchema namespace.

Example:

None.

G1737

Statement:

Define a target namespace in schemas.

Rationale:

A target namespace describes the namespace for all the schema components defined by the schema. Without a target namespace, all enclosed schema components are not associated with a namespace and if a namespace prefix is not associated with the target namespace then all references to these schema components must be unqualified. By not specifying a target namespace, ambiguity can arise when the schema is integrated with other schemas. This can cause unnecessary naming collisions.

Note: *http://www.library.org is the target namespace as well the lib namespace. See the third targetNamespace line of the following code sample.*

```
<?xml version="1.0"?>
<xsd:schema
      targetNamespace="http://www.library.org"

      elementFormDefault="qualified">
 <xsd:include schemaLocation="BookCatalogue.xsd"/>
 <xsd:element name="Library">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="BookCatalogue">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element ref="lib:Book"
                            maxOccurs="unbounded"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Referenced By:

Using XML Namespaces
Design Tenet: Open Architecture
Interoperability
Design Tenet: Make Data Interoperable
Design Tenet: Make Data Understandable

Evaluation Criteria:

1) Test: [G1737.1]

Does the schema declare a target namespace?

Procedure:

Check the definition of all schemas and look for the assignment of the targetNamespace attribute.

Example:

```
<xsd:schema

  targetNamespace="http://www.library.org"
 >
  . . .
</xsd:schema>
```

G1738

Statement:

Define a qualified namespace for the target namespace.

Rationale:

To force all schema components defined by the schema to be qualified and to belong to a namespace, associate a qualified namespace with the target namespace. This causes all components defined within the namespace to be explicitly associated with a namespace. In other words, all components are always qualified.

Note: *http://www.library.org is the target namespace as well the lib namespace. See the forth xmlns:lib line of the following code sample.*

```
<?xml version="1.0"?>
<xsd:schema
      targetNamespace="http://www.library.org"

      elementFormDefault="qualified">
 <xsd:include schemaLocation="BookCatalogue.xsd"/>
 <xsd:element name="Library">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="BookCatalogue">
         <xsd:complexType>
           <xsd:sequence>
             <xsd:element ref="lib:Book"
                           maxOccurs="unbounded"/>
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
     </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Using XML Namespaces

Evaluation Criteria:

1) Test: [G1738.1]

Does the schema declare a qualified namespace for the target namespace?

Procedure:

Check the definition of all schemas and look for the assignment of the targetNamespace attribute and make sure there is also a qualified namespace with the same name.

Example:

In this example, the targetNamespace and the qualified namespace lib both have the same URI associated with them.

```
<xsd:schema

  targetNamespace="http://www.library.org"
 >
  . . .
</xsd:schema>
```

G1740

Statement:

Append the suffix Type to XML type names.

Rationale:

Syntactically, XML allows names within a namespace to be reused as long as they do not define the same XML Schema component. Therefore, a type and an element can both have the same name. A parser can easily differentiate the components, but a human can not. In order to maintain maintainable "user-friendly" code, differentiate types and elements by adding a type suffix for types.

Referenced By:

Defining XML Types
Maintainability

Evaluation Criteria:

1) Test: [G1740.1]

Do all the complex type names end in the type suffix?

Procedure:

Examine all the complex and simple type schema component definitions and verify that they end in the suffix type.

Example:

None.

G1744

Statement:

Only reference abstract **XML elements** in substitution groups.

Rationale:

An abstract **XML element** can not have its type instantiated in an instance document. This means that the element used as the basis for the substitution group and all the members of the substitution group must be derived from the same type.

Referenced By:

Maintainability
Using XML Substitution Groups

Evaluation Criteria:

1) Test: [G1744.1]

Is the element used as the basis for the substitution group declared to be abstract and is it derived from a type?

Procedure:

Examine all the elements used as the basis for substitution groups and verify that they have been declared as abstract.

Example:

```
<xsd:element name="RecordingMedium"
      abstract="true"
      type="RecordingMediumType"/>
```

G1745

Statement:

Append the suffix Group to substitution group **XML element** names.

Rationale:

Syntactically, XML allows names within a namespace to be reused as long as they do not define the same XML Schema component. Therefore, a type and an **XML element** can both have the same name. A parser can easily differentiate the components, but a human can not. In order to maintain maintainable "user-friendly" code, differentiate types and elements by adding a type suffix for types.

Referenced By:

Using XML Substitution Groups
Maintainability

Evaluation Criteria:

1) Test: [G1745.1]

Do all the complex type names end in the type suffix?

Procedure:

Examine all the complex and simple type schema component definitions and verify that they end in the suffix type.

Example:

None.

G1746

Statement:

Develop XSLT stylesheets that are XSLT version agnostic.

Rationale:

There are never any guarantees as to the XSLT environment that a stylesheet will be used in. There are ways of writing code as recommended by the W3C so that the stylesheets operate in XSL Version 1.0, 2.0 and future releases. See W3C Extensibility and Fallback for XSL Transformations (XSLT) 2.0 for details.

Referenced By:

Design Tenet: Make Data Interoperable
XSLT
Design Tenet: Open Architecture
Interoperability

Evaluation Criteria:

1) Test: [G1746.2]

Does the stylesheet support 2.0 and future version portability as defined by the W3C Extensibility and Fallback for XSL Transformations (XSLT) 2.0?

Procedure:

Look for the use of the use-when attribute in the xsl:value element.

Example:

```
<xsl:value-of
   select="pad($input, 10)"
   use-when="function-available('pad', 2)"
/>
<xsl:value-of
  select
    ="concat
       ( $input,
         string-join
          ( for $i in
              1 to
              10 - string-length($input)
              return ' ',
           ''
          )
       )"
  use-when="not(function-available('pad', 2)
"/>
```

2) Test: [G1746.1]

Does the stylesheet support version 1.0 and 2.0 portability as defined by the W3C Extensibility and Fallback for XSL Transformations (XSLT) 2.0?

Procedure:

Look for the use of the xsl:when and xsl:otherwise construct where the 2.0 functions are tested for availability in the xsl:when branch and the 1.0 functionality is defined in the xsl:otherwise branch. For a comprehensive list of 2.0 functions see the W3Schools site on XPath, XQuery and XSLT Functions.

Example:

```
<out xsl:version="2.0">
 <xsl:choose>
   <xsl:when
      test="function-available('matches')">
     <xsl:value-of
        select="matches($input, '[a-z]*')"/>
   </xsl:when>
   <xsl:otherwise>
     <xsl:value-of
        select=
          = "string-length
              ( translate
               ( $in,
                 'abcdefghijklmnopqrstuvwxyz',
                   ''
               )
              )
            = 0"
      />
   </xsl:otherwise>
 </xsl:choose>
</out>
```

G1751

Statement:

Document all XSLT code.

Rationale:

XSLT is source code and should be internally documented including a file header that describes the purpose of the transform and any restrictions or caveats associated with the transform.

Referenced By:

Maintainability
XSLT

Evaluation Criteria:

1) Test: [G1751.1]

Doe the XSLT have internal comments that document the transform?

Procedure:

Look inside the XSLT code and look for internal comments.

Example:

```
<xsl:for-each
  select="/transactions/transaction">
 <!--
    NOTE: Since dates are currently in
    ISO format they are in a sorted format
    and need no multi-level sorting
  -->
 <xsl:sort
    order="ascending"
    select="@startdate"/>
 <tr>
   <td>
     <xsl:value-of
        select="@startdate"/>
   </td>
   <td>
     <xsl:value-of
        select="@description"/>
   </td>
   <td>
     <!#  Get year
           1234567890
           yyyy/mm/dd
      -->
     <xsl:value-of
        select="substring(@startdate, 1,4)"
      />
   </td>
   <td>
     <!#  Get month
           1234567890
           yyyy/mm/dd
      -->
     <xsl:value-of
        select="substring(@startdate, 6,2)"/>
   </td>
   <td>
     <!#  Get day
           1234567890
           yyyy/mm/dd
      -->
     <xsl:value-of
        select="substring(@startdate, 9,2)"/>
   </td>
 </tr>
</xsl:for-each>
```

G1753

Statement:

Declare the XML schema version with an **XML attribute** in the root **XML element** of the schema definition.

Rationale:

Formalizing the schema version number through the use of a required **XML attribute** helps automate the process of validating the versions. This will reduce unexpected runtime errors that occur when assumptions are made about the schema that may change over time. (See http://www.xfront.com/SchemaVersioning.html)

Referenced By:

Interoperability
Versioning XML Schemas
Design Tenet: Make Data Understandable
Design Tenet: Open Architecture
Design Tenet: Make Data Interoperable
Maintainability
Design Tenet: Provide Data Management

Evaluation Criteria:

1) Test: [G1753.1]

Does the schema definition define a required attribute that captures the version information?

Procedure:

Look at the schema definition file and look for the inclusion of a required attribute that captures the schema version number. In the following example, the schemaVersion attribute is defined.

Example:

```
<xs:schema

  targetNamespace="http://www.exampleSchema"
  xmlns: xs ="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1.3"
>
 <xs:element name="Example">
   <xs:complexType>
      . . .
     <xs:attribute
        name="schemaVersion"
        type="xs:decimal"
        use="required"
      />
   </xs:complexType>
 </xs:element>
```

G1754

Statement:

Give each new XML schema version a unique URL.

Rationale:

This allows the previous versions of the schema to be made available to support uninterrupted processing and supports an orderly transition. It also allows the users of the schemas to compare and contrast the evolving schema. http://www.xfront.com/SchemaVersioning.html

Referenced By:

Design Tenet: Open Architecture
Design Tenet: Make Data Interoperable
Maintainability
Versioning XML Schemas
Interoperability

Evaluation Criteria:

1) Test: [G1754.1]

Look for the multiple schemas that represent different versions with different URLs.

Procedure:

Look for XSDs that all define a particular schema but can be found at different locations. This can be done by changing the path to the schema definition or that change the name of the file by adding the version number.

Example:

Changing the file path:

```
http://www.some.org/schema/1999/CoiSchema
http://www.some.org/schema/2003/CoiSchema
http://www.some.org/schema/2006/CoiSchema
```

Changing the file name:

```
http://www.some.org/schema/CoiSchema_1999
http://www.some.org/schema/CoiSchema_2003
http://www.some.org/schema/CoiSchema_2006
```

G1756

Statement:

Isolate XPath expression statements into the configuration data.

Rationale:

XPath expression statements are dependent on the XML Schemas that are associated with the documents. Consequently they need maintained independently from the applications that use them. Storing the XPath expression statements externally as part of the configuration data ensures a clean separation of the maintenance tasks and supports traceability using configuration management tools.

Referenced By:

XPath
Maintainability

Evaluation Criteria:

1) Test: [G1756.1]

Are there XPath expression statements embedded as string literals in the application source code?

Procedure:

Look for the occurrence of XPath expression statements or XML Element names defined as strings within the source code.

Example:

```
void main ( String args)
{ . . .
  String titleSearchExpression
    = "/library/books/book/title";
  . . .
} // End main
```

G1759

Statement:

Use a style guide when developing Web portlets.

Rationale:

Portals contain portlets from different sources, and it is important for usability for the portal to have a common look and feel across all portlets.

Referenced By:

Design Tenet: Make Data Interoperable
Interoperability
Design Tenet: Make Data Understandable
Reusability
Human Factor Considerations for Web-Based User Interfaces

Evaluation Criteria:

1) Test: [G1759.1]

Do all portlets comply with a style guide.

Procedure:

Look at development documentation to determine if a style guide exist for web portlets and look for code reviews that show it was used during development.

Example:

- Ahlstrom, V. & Allendoerfer, K. Web-Based Portal Computer-Human Interface Guidelines, 2004. Retrieved from: http://hf.tc.faa.gov/products/bibliographic/tn0423.htm (July 2006).

- Web Portal Design Guide , Fernandes, K., Space and Nabal Warfare Systems Center San Diego 2006

G1760

Statement:

Solicit feedback from users on user interface usability problems.

Rationale:

Active testing and solicitation of input from users helps identify usability problems with the user interface and helps to identify areas that may reduce performance or require excessive cognitive attention by the user.

Referenced By:

Human-Computer Interaction
Design Tenet: Be Responsive to User Needs

Evaluation Criteria:

1) Test: [G1760.1]

Does the program solicit user feedback for user interface usability problems?

Procedure:

Determine if user surveys are conducted on the usability of the system.

Example:

G1761

Statement:

Provide units of measurements when displaying data.

Rationale:

Displayed units for measurable data provide for better understanding the data and enable reuse of the data. (This guidance is derived from MIL-STD 1472F)

Referenced By:

Design Tenet: Make Data Interoperable
Human-Computer Interaction
Interoperability
Design Tenet: Make Data Understandable

Evaluation Criteria:

1) Test: [G1761.1]

Does the system display units for all measurable data?

Procedure:

Inspect the user interfaces for system and check that units are shown for all measurable data.

Example:

Length displayed as meters
Distance displayed as miles.

G1762

Statement:

Indicate all simulated data as simulated.

Rationale:

Simulated data that is not marked as simulated may be of misinterpreted and can decrease system, user, or system safety. (This guidance is derived from MIL-STD 1472F)

Referenced By:

Design Tenet: Make Data Trustable
Human-Computer Interaction
Design Tenet: Make Data Understandable

Evaluation Criteria:

1) Test: [G1762.1]

Is all simulated data clearly marked as simulated?

Procedure:

Check system inputs and outputs including user interfaces and check that the simulated data is properly labeled as simulated.

Example:

None.

G1763

Statement:

Indicate the security classification for all classified data.

Rationale:

Displaying classified data without clearing marking the classification can lead to incorrect assumptions about the data. This can lead to improperly use of the data or prevent the data from being reused due to lack of clear understanding of the classification. (This guidance is derived from MIL-STD 1472F)

Referenced By:

Interoperability
Design Tenet: Make Data Accessible
Design Tenet: Make Data Understandable
Design Tenet: Make Data Trustable
Human-Computer Interaction
Design Tenet: Make Data Interoperable

Evaluation Criteria:

1) Test: [G1763.1]

Does the system display classification markings for all classified data?

Procedure:

Check the system outputs and user interfaces for classification marking for all classified data or systems.

Example:

Classification banners on monitors
Classification banners on printouts

G1770

Statement:

Explicitly define the **Data Distribution Service** (DDS) **Domains** for the system.

Rationale:

DDS uses Domains to separate the **Global Data Spaces** into independent areas. **Topics** written to one DDS Domain are completely hidden from the other DDS Domains. Use DDS Domains for isolation (hiding subsystem data from other parts of the system), modularity, and scalability. In order for systems to benefit from these advantages, they must explicitly define their own DDS Domains rather than use the default DDS Domain.

Referenced By:

DDS Domains - Global Data Spaces
Design Tenet: Make Data Interoperable
Design Tenet: Open Architecture
Interoperability
Design Tenet: Make Data Understandable

Evaluation Criteria:

1) Test: [G1770.1]

Is the system using different **`DomainId`** values to isolate the subsystems?

Procedure:

Look for multiple calls to **`create_participant()`** operation on the **`DomainParticipantFactory`**.

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
realtimeQuoterParticipant
  = participantFactory->create_participant
      ( REALTIME_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

G1771

Statement:

Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of a **publisher**.

Rationale:

DDS relies on the use of QoS characteristics to match publishers with **subscribers**. If the publishers do not specify a QoS policy other than the default, much of the power of DDS publishing is lost and the capabilities of the publisher are not documented.

Referenced By:

DDS Quality of Service
Design Tenet: Differentiated Management of Quality-of-Service
Interoperability

Evaluation Criteria:

1) Test: [G1771.2]

Is the **`get_default_publisher_qos`** operation used to create publisher?

Procedure:

Look for the use of the **`get_default_publisher_qos`** operation within the code.

Example:

```
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
DDS::PublisherQos publisherQos;
Participant->get_default_publisher_qos
  ( publisherQos );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

2) Test: [G1771.1]

Are values other than the **`PUBLISHER_QOS_DEFAULT`** value used to create publishers?

Procedure:

Verify that the **`PUBLISHER_QOS_DEFAULT`** constant is not used within the code.

Example:

```
DDS::Publisher publisher
  = participant->create_publisher
      ( PUBLISHER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

G1772

Statement:

Assign a unique identifier for each **Data-Distribution Service** (DDS) **Domain** within the system.

Rationale:

DDS uses Domains to separate the **Global Data Spaces** into independent areas. Within DDS, a unique identifier called the **`DomainId`** identifies each DDS Domain.

Referenced By:

DDS Domains - Global Data Spaces
Design Tenet: Make Data Interoperable
Interoperability

Evaluation Criteria:

1) Test: [G1772.1]

Is there a single value for the **`DomainId`** used for each Domain when the **`create_participant`** operation is used?

Procedure:

Look for the use of the **`create_participant`** operation within the code.

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
realtimeQuoterParticipant
  = participantFactory->create_participant
      ( REALTIME_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111

G1773

Statement:

Use **`#include`** guards for all headers.

Rationale:

Including a guard prevents including a header file more than once. There are two basic kinds of guards: internal and external. Internal guards occur in each header file that is to be included. External guards occur in a file that includes a header file. In the past, there were compiling performance issues using internal guards because the file had to be scanned each time the file was included. This has been optimized away by most modern compilers. Furthermore, external guards are fragile and tightly coupled since the file including the header and header file must use the same guard name.

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 24.*

Referenced By:

Reusability
C++ Header Files
Maintainability

Evaluation Criteria:

1) Test: [G1773.1]

Do all header files contain include guards?

Procedure:

Check each file that is included using a **`#include`** statement to make sure it has an include guard.

Example:

An internal guard looks like this:

G1774

Statement:

Make header files self-sufficient.

Rationale:

To enable code reuse, each unit of code should be able to be compiled independently without having to follow a predetermined build order or having to know the dependencies. Code is difficult to reuse when the dependencies are not clearly documented. Therefore, ensure each header is capable of being used by itself (i.e, it can be compiled standalone) by having it include all the headers upon which it depends.

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 23.*

Referenced By:

Maintainability
C++ Header Files
Reusability

Evaluation Criteria:

1) Test: [G1774.1]

Can each class be compiled by itself without having to compile other units?

Procedure:

Compile each class as a standalone file and check compile output for errors caused by missing definitions.

Example:

None

G1775

Statement:

Do not overload the logical **`AND`** operator.

Rationale:

The logical **`AND`** operator has a special relationship with the compiler. When a logical **`AND`** operator is written to overload the inherent operators, the precedence of operation (i.e., left side of operator or right side of operator) is undefined. This can result in compiler dependency. In the following code, it is not clear whether the **`DisplayPrompt`** will execute first or the **`GetLine`** operation will executed first.

```
if ( DisplyPrompt() && GetLine() )
```

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 30.*

Referenced By:

Reusability
Maintainability
C++ Operator Overloading

Evaluation Criteria:

1) Test: [G1775.1]

Is the logical **`AND`** operator defined?

Procedure:

Look for the overloading of the logical **`AND`** operator.

Example:

None

G1776

Statement:

Do not overload the logical **`OR`** operator.

Rationale:

The logical **`OR`** operator has a special relationship with the compiler. When a logical **`OR`** operator is written to overload the inherent operators, the precedence of operation (i.e., left side of operator or right side of operator) is undefined. This can result in compiler dependency.

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 30.*

Referenced By:

C++ Operator Overloading
Reusability
Maintainability

Evaluation Criteria:

1) Test: [G1776.1]

Is the logical **`OR`** operator defined?

Procedure:

Look for the overloading of the logical **`OR`** operator.

Example:

None

G1777

Statement:

Do not overload the **`comma`** operator.

Rationale:

The **`comma`** operator has a special relationship with the compiler. When a **`comma`** operator is written to overload the inherent operators, the precedence of operation (i.e., left side of operator or right side of operator) is undefined. This can result in compiler dependency.

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 30.*

Referenced By:

C++ Operator Overloading
Maintainability
Reusability

Evaluation Criteria:

1) Test: [G1777.1]

Is the **`comma`** operator defined?

Procedure:

Look for the overloading of the **`comma`** operator.

Example:

None

G1778

Statement:

Place all **`#include`** statements before all namespace **`using`** statements.

Rationale:

Files that are included can contain their own **`using`** clauses. In order to make sure that the **`using`** statements are not overridden by these subsequent using definitions, place all using statements after all include statements.

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 59.*

Referenced By:

C++ Namespaces and Modules
Reusability
Maintainability

Evaluation Criteria:

1) Test: [G1778.1]

Are all the **`using`** statements defined after all the **`#include`** statements?

Procedure:

Scan all files and make sure that all the **`using`** statements occur after all **`using`** statements.

Example:

None

G1779

Statement:

Explicitly namespace-qualify all names in header files.

Rationale:

Header files are meant to be included by other files. A header file inclusion should not alter the meaning of code that it is included in as this behavior is unexpected. Therefore, use fully-qualified names in header files and do not use using directives or declarations. This also promotes clarity in the header file whose main purpose is to communicate the interface to the implementation class.

Note: *This practice has been adapted from Sutter and Alexandrescu, standard practice 59.*

Referenced By:

C++ Header Files
Reusability
C++ Namespaces and Modules
Maintainability

Evaluation Criteria:

1) Test: [G1779.1]

Are named fully namespace qualified throughout the header files?

Procedure:

Scan all header files and make sure that all namespaces are fully qualified.

Example:

None

2) Test: [G1779.2]

Are all header files free from using directives or declarations?

Procedure:

Scan all header files to determine that they do not contain using directives or declarations.

Example:

None

G1784

Statement:

Include a statement in the solicitation for Contractors to identify and list data rights for all proposed products.

Rationale:

Reusing GOTS requires understanding all the data rights associated with each artifact involved with the solution.

Referenced By:

Section K: Representations, Certifications, and Other Statements of Offerors (Data Rights)
Reusability

Evaluation Criteria:

1) Test: [G1784.1]

Does the solicitation include a statement for the offerer
to identify data rights for all proposed products?

Procedure:

Review the solicitation and identify statements that
require the offerer to identity data rights for all proposed
products.

Example:

Example data rights markings include markings for Unlimited Rights and Government Purpose Rights.

G1785

Statement:

Stipulate that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on reuse of common functionality.

Rationale:

The Government must stipulate what evaluation criteria will be used to evaluate proposed solutions. Having the Offeror specify the extent to which proposed solutions build on reuse of common functionality aids in the evaluation of proposals and aids in identification of common functionality.

Referenced By:

Reusability
Section M: Evaluation Factors for Award
Interoperability

Evaluation Criteria:

1) Test: [G1785.1]

Has the government stipulated that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on reuse of common functionality?

Procedure:

Check Section M for a statement that states reuse of common functionality will be used as an evaluation criterion for proposals.

Example:

None.

G1786

Statement:

Stipulate that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on well defined services.

Rationale:

The Government must stipulate what evaluation criteria will be used to evaluate proposed solutions. Having the Offeror specify the extent to which proposed solutions build on reuse of well defined services aids in the evaluation of proposals and further improves service reuse.

Referenced By:

Section M: Evaluation Factors for Award
Interoperability
Reusability

Evaluation Criteria:

1) Test: [G1786.1]

Has the government stipulated that evaluation criteria will include the extent to which an Offeror's proposed technical solution builds on well defined services?

Procedure:

Check Section M for a statement that states the extent to which the proposed solution builds on well defined services will be used as an evaluation criterion for proposals.

Example:

None.

G1787

Statement:

Stipulate that the Offeror is to use NESI to assess net-centricity and interoperability.

Rationale:

NESI guidance and its associated checklists are useful tools (used by themselves or in conjunction with other tools) for assessing how a program is meeting its net-centric and interoperability objectives.

Referenced By:

Interoperability
Reusability
Post Award Contract Actions
Section J: List of Attachments

Evaluation Criteria:

1) Test: [G1787.1]

Has the Government stipulated that the Offeror is to use NESI to assess net-centricity and interoperability?

Procedure:

Identify statements in policy, RFPs, SOWs, or CDRLs that stipulate that the Offeror is to use NESI to assess net-centricity and interoperability?

Example:

PEO C4I uses the Technical Evaluation Checklist (http://nesipublic.spawar.navy.mil/checklist) as a means for Program Managers to assess how well their programs meet net-centric objectives.

G1788

Statement:

Stipulate that the Offeror is to use Government approved data rights labels and markings for all deliverables that are identified as Unlimited or Government Purpose Rights.

Rationale:

Reusing deliverables or components of deliverables requires a full understanding of the data rights associated with each artifact in the deliverable. Identified data rights for each artifact through the use of data right labels are important in order to protect the legal rights of both the contractor and government during component reuse.

Referenced By:

Section J: List of Attachments
Reusability
Post Award Contract Actions

Evaluation Criteria:

1) Test: [G1788.1]

Has the government stipulated that the Offeror is to use government approved data rights labels and markings for all deliverables that are identified as Unlimited or Government Purpose Rights.

Procedure:

Identify statements in the RFP, SOW, or CDRLs which mandate the use of government approved data rights labels for any deliverables that are identified as Unlimited or Government Purpose Rights.

Example:

None.

G1796

Statement:

Explicitly define all the **Data Distribution Service** (DDS) **Domain Topics**.

Rationale:

DDS uses Topics to define the information model. Topics are identified by an application-defined string and an associated **data type**. Topics represent collections of object sin the **Global Data Space**; individual data-objects within a Topic are identified by the value of the key fields which are some special fields inside the data-type. Applications use Topics to publish the information and subscribe to the information they want.

In a DDS system information exchange happens as a result of **publishers** and **subscribers** agreeing to use the same Topics. Therefore the selection of the Topic names and their semantic meaning is an important part of system design.

Referenced By:

Messaging within a DDS Domain
Design Tenet: Make Data Interoperable
Interoperability
Design Tenet: Make Data Understandable

Evaluation Criteria:

1) Test: [G1796.1]

Are all the Topics (and Topic names) the system uses explicitly defined and captured in a centralized document (e.g., Excel table, XML file, dedicated tool)?

Procedure:

Look for documentation that contains listings for all Topics the system uses.

Example:

```
<topic>
 <name>Temperature</name>
 <type>TemperatureData</type>
 <description>
   This topic contains a reading of
   a temperature sensor
 </description>
</topic>
<topic>
  . . .
</topic>
```

G1797

Statement:

Use a minimum of 1024 bits for **asymmetric keys**.

Rationale:

Strong encryption helps to prevent unauthorized data decryption using modern day resources.

Referenced By:

Design Tenet: Identity Management, Authentication, and Privileges
Interoperability
Encryption Services
Design Tenet: Encryption and HAIPE

Evaluation Criteria:

1) Test: [G1797.1]

Are asymmetric key encryption levels at least 1024 bit?

Procedure:

Check the server configuration and verify that the asymmetric keys being used are at least 1024 bit.

Example:

Verified Web server ciphers under the SSL portion of the configuration pages of the administration server.
For Internet Explorer 5.0 and above, click the **`Help`**Help menu and then click the **`About Internet Explorer`** option. The **`About`** box will list the Cipher Strength.

2) Test: [G1797.2]

Is the application using domestic (U.S.) grade ciphers?

Procedure:

Verify that the application supports domestic (U.S.) grade ciphers.

Example:

None.

G1798

Statement:

Explicitly define all the **Data Distribution Service** (DDS) **Domain data types**.

Rationale:

DDS provides support for writing and reading typed data. For each application data type, DDS creates the necessary objects that allow manipulation of the data object. For example, for a given data type named **`MyDT`**, DDS creates a **`MyDTDataWriter`** and **`MyDTDataReader`**.

Knowing the data type of the object allows DDS to marshal the data properly. Consequently, any computer platform and/or language can process the data properly . For example, DDS performs the proper endianess transformations, alignment, and adjustment for 32 versus 64 bit platforms.
Knowing the data type is also required for the proper functioning of **`ContentFilteredTopics`**.

Moreover, explicit definition of the data types is required for the tools provided by DDS vendors to display and manipulate the data properly. Visualization tools, logging and replay, automatic bridging to other middleware, etc., all depend on data type transparency.

Referenced By:

Messaging within a DDS Domain
Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Interoperability

Evaluation Criteria:

1) Test: [G1798.1]

Are all the data types the system uses explicitly defined using IDL which is either manually written or generated from equivalent UML or XML representations?

Procedure:

Look for the IDL (or equivalent XML) files used to define the types used by the system.

Example:

```
// File MyTpes.idl
struct MyType
{
   long x;
   long y;
   string<10> units;
};
```

G1799

Statement:

Explicitly associate data types to the **Data Distribution Service** (DDS) **Topics** within a DDS **Domain**

Rationale:

A DDS Topic represents a homogeneous collection of data-objects in the **Global Data Space**. All data-objects within a Topic share a common **data-type**. Knowledge of the type associated with the Topic is required for an application to be able to publish and subscribe data on the Topic.

Referenced By:

Interoperability
Messaging within a DDS Domain
Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable

Evaluation Criteria:

1) Test: [G1799.1]

Do all Topics have an explicit association to a data type.

Procedure:

Look for documentation that lists the Topics in use by the system and verify that each Topic has a data type associated with it

Example:

```
<topic>
 <name>Temperature</name>
 <type>TemperatureData</type>
 <description>
    This topic contains a reading of
    a temperature sensor
 </description>
</topic>
<topic>
  . . .
</topic>
```

G1800

Statement:

Explicitly identify Keys within the **Data Distribution Service** (DDS) **data type** that uniquely identify an instance of a data object.

Rationale:

Within each DDS **Domain** (i.e., **Global Data Space**) a data-object is identified by the tuple (**Topic**, Key). The Key is a set of fields within the data type associated with the Topic that the application has tagged to indicate their role in uniquely identifying the data object. For example, if the Topic represents a person to the IRS, the Key may be simply the field containing the social security number.

The proper definition of the key is necessary to allow DDS to implement the **`KEEP_LAST HISTORY`** QoS properly as well as to enforce QoS policies such as **`DEADLINE`**, and **`OWNERSHIP`**. It is also necessary in order for DDS to supply the proper Sample information to the **DataReader**.

All data types require Keys except in the case where the Topic logically represents a single object, for example when the Topic represents a Message Queue.

Referenced By:

Messaging within a DDS Domain
Design Tenet: Make Data Interoperable
Design Tenet: Make Data Understandable
Interoperability

Evaluation Criteria:

1) Test: [G1800.1]

Does the declaration of the data-type associated with the Topic explicitly designate using one or more of the fields as a Key?

Procedure:

Examine the IDL (or equivalent XML) files used to define the types used by the system to identify the declaration of the data-type associated with each Topic (i.e., see if there are any tags that designate which fields form the Key).

Example:

```
For data types defined using IDL:
struct SensorData
{
  long    sensor_id; //@key
  float   value;
  string<32> units;
  string<64> location;
};
struct DepartingFlightData
{
  string<8>    airline_code;  //@key
  long         flight_number; //@key
  string<8>    destination_airport_code;
  string<2>    departing_terminal;
  long         departing_gate;
  FlightTime   scheduled_departure_time;
  FlightTime   expected_departure_time;
  string<32>   status;
};
```

G1801

Statement:

Explicitly define a **Topic Quality of Service** (QoS) for each **Data Distribution Service** (DDS) Topic within a DDS **Domain**.

Rationale:

DDS Topics define the information model of the system. The QoS Policies associated with the Topics define expectations and constraints that all users (**publishers** or **subscribers**) of the Topic should know. Consequently, definition of the Topic QoS is an important part of the system design.

Referenced By:

Interoperability
Messaging within a DDS Domain
Design Tenet: Differentiated Management of Quality-of-Service
DDS Quality of Service

Evaluation Criteria:

1) Test: [G1801.1]

Is there a document that defines the QoS Policies that each Topic uses and does the document that describes the Topics and their associated data types also provide information on the Topic QoS?

Procedure:

Look at the documents that define the Topics in use and their associated data-types and see if they also define the Topic QoS.

Example:

```
Topic: DepartingAircraft
Type: DepartingAircraftStruct
QoS: HISTORY kind=KEEP_LAST
QoS: RELIABILITY kind=RELIABLE
QoS: DEADLINE duration=15minutes
QoS: LIFESPAN duration = 1 hour
Etc.
```

G1803

Statement:

Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe real-time messaging criteria for **Publishers**.

Rationale:

DDS relies on the use of a QoS set of characteristics to match publishers with **subscribers**. If the publishers do not specify a QoS policy other than the default, much of the power of DDS publishing is lost and the capabilities of the publisher are not documented.

Referenced By:

DDS Quality of Service
Design Tenet: Differentiated Management of Quality-of-Service
Interoperability

Evaluation Criteria:

1) Test: [G1803.2]

Is the **`get_default_publisher_qos`** operation used to create publisher?

Procedure:

Look for the use of the **`get_default_publisher_qos`** operation within the code.

Example:

```
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
DDS::PublisherQos publisherQos;
Participant->get_default_publisher_qos
  ( publisherQos );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test: [G1803.1]

Is the **`PUBLISHER_QOS_DEFAULT`** value used to create publishers?

Procedure:

Look for the use of the **`PUBLISHER_QOS_DEFAULT`** constant within the code.

Example:

```
DDS::Publisher publisher
  = participant->create_publisher
      ( PUBLISHER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

G1804

Statement:

Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe **DataWriter**.

Rationale:

DDS relies on the use of QoS characteristics to match a **`DataWriter`** with each **DataReader** of the same **Topic**. If the **`DataWriter`** does not specify a QoS policy other than the default, much of the power of DDS publishing is lost and the capabilities of the **`DataWriter`** are not documented.

Referenced By:

Design Tenet: Differentiated Management of Quality-of-Service
Interoperability
DDS Quality of Service

Evaluation Criteria:

1) Test: [G1804.2]

Is the **`get_default_datawriter_qos`** operation used to create participant?

Procedure:

Look for the use of the **`get_default_datawriter_qos`** operation within the code.

Example:

```
DDS::DataWriterQos dataWriterQos;
publisher->get_default_datawriter_qos
  ( dataWriterQos );
DDS::DataWriter dataWriter
  = publisher ->create_datawriter
      ( myTopic,
        dataWriterQos,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test: [G1804.1]

Is the **`DATAWRITER_QOS_DEFAULT`** value used to create **`DataWriter?`**

Procedure:

Look for the use of the **`DATAWRITER_QOS_DEFAULT`** constant within the code.

Example:

```
DDS::DataWriter dataWriter
```

```
  = participant->create_datawriter
      ( myTopic,
        DATAWRITER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

G1805

Statement:

Explicitly define the **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of the **Subscriber**.

Rationale:

DDS relies on the use of QoS set of characteristics to match subscribers with **publishers**. If the subscribers do not specify a QoS policy other than the default, much of the power of DDS subscription and publishing is lost and the requirements of the subscriber are not documented.

Referenced By:

Design Tenet: Differentiated Management of Quality-of-Service
Interoperability
DDS Quality of Service

Evaluation Criteria:

1) Test: [G1805.1]

Is the **SUBSCRIBER_QOS_DEFAULT** value used to create subscribers?

Procedure:

Look for the use of the **SUBSCRIBER_QOS_DEFAULT** constant within the code.

Example:

```
DDS::Publisher publisher
  = participant->create_subscriber
      ( SUBSCRIBER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

DDS::STATUS_MASK_ALL is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test: [G1805.2]

Is the **get_default_subscriber_qos** operation used to create subscribers?

Procedure:

Look for the use of the **get_default_subscriber_qos** operation within the code.

Example:

```
participant
  = participantFactory->create_participant
      ( QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
DDS::SubscriberQos subscriberQos;
Participant->get_default_subscriber_qos
  ( subscriberQos );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

G1806

Statement:

Explicitly define the Request-Offered **Data Distribution Service** (DDS) **Quality of Service** (QoS) Policies to describe the behavior of the **DataReader**.

Rationale:

DDS relies on the use of QoS characteristics to match a **DataWriter** with each **`DataReader`** of the same Topic. If the **`DataReader`** does not specify a QoS policy other than the default, much of the power of DDS subscription and publishing is lost and the requirements of the **`DataReader`** are not documented.

Referenced By:

Interoperability
Design Tenet: Differentiated Management of Quality-of-Service
DDS Quality of Service

Evaluation Criteria:

1) Test: [G1806.1]

Is the **`DATAREADER_QOS_DEFAULT`** value used to create **`DataReader`**?

Procedure:

Look for the use of the **`DATAREADER_QOS_DEFAULT`** constant within the code.

Example:

```
DDS::DataResder dataReader
  = participant->create_datareader
      ( DATAREADER_QOS_DEFAULT,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

2) Test: [G1806.2]

Is the **`get_default_datareader_qos`** operation used to create participant?

Procedure:

Look for the use of the **`get_default_datareader_qos`** operation within the code.

Example:

```
DDS::DataReaderQos dataReaderQos;
publisher->get_default_datareader_qos
  ( dataReaderQos );
DDS::DataReader dataReader
```

```
  = publisher ->create_datareader
      ( myTopic,
        dataReaderQos,
        NULL,
        DDS::STATUS_MASK_ALL
      );
```

G1808

Statement:

Handle all **Data Distribution Service** (DDS) **Quality of Service** (QoS) contract violations using one of the **Subscriber access APIs**.

Rationale:

QoS contract violations typically indicate either a system mis-configuration, or else a transient failure (e.g., a network that has been temporarily disconnected). Either way the application must monitor these events to determine if they are relevant to their operation and consequently take proper corrective action.

Referenced By:

Interoperability
Design Tenet: Differentiated Management of Quality-of-Service
DDS Quality of Service

Evaluation Criteria:

1) Test: [G1808.1]

Are all the DDS QoS-related status change events are captured via a DDS **Listener** or a DDS **WaitSet**?

Procedure:

Specifically ensure that the following DDS events are handled. Look at the arguments passed to the **`create_domain_participant`**, **`create_datawriter`**, and **`create_datareader_operations`** and check that the listener and mask parameters to verify that the following events are being handled:

- OFFERED_DEADLINE_MISSED_STATUS
- REQUESTED_DEADLINE_MISSED_STATUS
- OFFERED_INCOMPATIBLE_QOS_STATUS
- REQUESTED_INCOMPATIBLE_QOS_STATUS
- LIVELINESS_LOST_STATUS
- LIVELINESS_CHANGED_STATUS

Example:

```
participantFactory
  = TheParticipantFactory;
quickQuoterParticipant
  = participantFactory->create_participant
      ( QUICK_QUOTER_DOMAIN_ID,
        PARTICIPANT_QOS_DEFAULT,
        participantListener,
        DDS::STATUS_MASK_ALL
      );
```

`DDS::STATUS_MASK_ALL` is part of DDS 1.3, prior releases require application to use 0x11111111.

G1810

Statement:

Use **data models** to document the data contained within the **Data Distribution Service** (DDS) **Data-Centric Publish Subscribe** (DCPS).

Rationale:

DCPS contains static and raw data that can be used is any number of views or objects. As a consequence, changes in the definition of the data, its DDS **Domains** or its structure can have a huge cascading effect. To minimize the impact of these changes, data needs to be documented in a data model that is not subject to implementation.

Referenced By:

Design Tenet: Make Data Understandable
Design Tenet: Make Data Interoperable
Reading/Writing Objects within a DDS Domain
Interoperability
Decoupling Using DDS and Publish-Subscribe

Evaluation Criteria:

1) Test: [G1810.1]

Is there a conceptual data model that captures the data within the DCPS?

Procedure:

Look for a data model that captures the data within the Data-Centric Publish-Subscribe (DCPS). The following is a very short list of some of the files extensions that may contain data models.

CDM	Conceptual model file (PowerDesigner)
PDM	Physical model file (PowerDesigner)
ER1	ERWin file
ERX	ERWin file
ERM	Entity Relationship Diagram Model file (Prosa)

Example:

G1835

Statement:

Document plans to migrate to **net-centricity**.

Rationale:

Net-centric migrations are often lengthy and subject to many factors. A formal migration plan guides the migration activities while addressing this complexity in an organized manner. Such a plan can provide tools for setting clear scope and targets, for measuring the migration progress against stated objectives, for proper documentation and for mitigating risks. Even small-scale migrations will benefit from having a formal migration plan, but the migration plan will be correspondingly less complex and easier to generate and maintain.

Referenced By:

Finalize Migration Plan
Migration Planning Process

Evaluation Criteria:

1) Test: [G1835.1]

Does the project have a plan to support migration to net-centricity?

Procedure:

Verify the presence of a plan supporting migration to net-centricity.

Example:

Ways to determine that there is a net-centric migration plan include reviewing the Information Support Plan (ISP) or contractual language (if the plan is a deliverable by contractor).

BP1255

Statement:

Use **surrogate keys**.

Rationale:

A surrogate key, also referred to as a system-generated key, database-sequence number, or arbitrary unique identifier, is a unique, arbitrary **primary key**. The **RDBMS** usually generates the surrogate key, but a database access layer such as the middle tier can also generate the surrogate key. The surrogate key is arbitrary because it is not derived from any data that exists within the table or the database. Another option for surrogate keys is Universally Unique Identifiers (UUIDs) (http://en.wikipedia.org/wiki/Universally_Unique_Identifier), the most common implementation being Microsoft's Globally Unique Identifiers (GUIDs) (http://en.wikipedia.org/wiki/Globally_Unique_Identifier).

Referenced By:

RDBMS Internals

BP1256

Statement:

Use surrogate keys as the **primary key**.

Rationale:

Instead of using the natural keys to identify each record uniquely, use a surrogate key. This allows the natural key information to be modified independently of the primary key and any foreign-key references to the key.

Referenced By:

RDBMS Internals
Database Development

Evaluation Criteria:

1) Test: [BP1256.1]

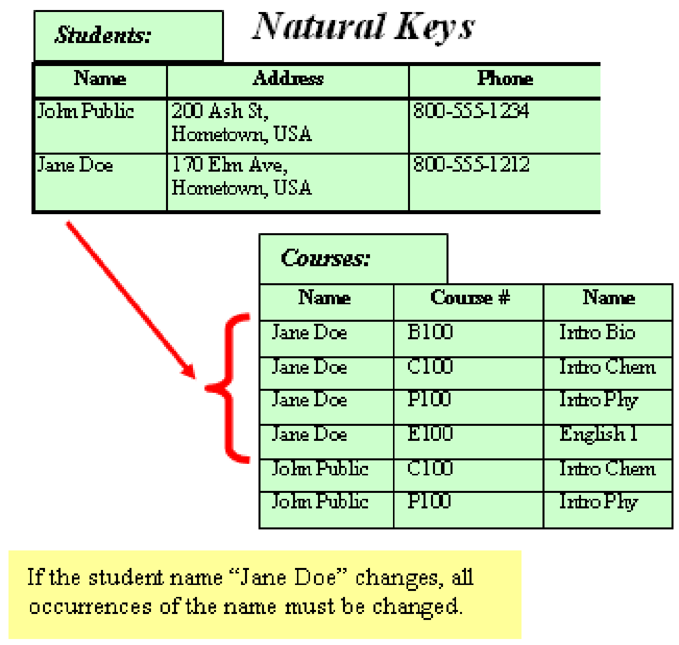
Are surrogate keys used instead of natural keys?

Procedure:

Look at the database metadata and determine if it uses surrogate or natural keys.

Example:

The following example shows natural keys. The primary keys are made up completely or in part from naturally occurring data in the tables.



Natural Keys

Students:

Name	Address	Phone
John Public	200 Ash St, Hometown, USA	800-555-1234
Jane Doe	170 Elm Ave, Hometown, USA	800-555-1212

Courses:

Name	Course #	Name
Jane Doe	B100	Intro Bio
Jane Doe	C100	Intro Chem
Jane Doe	P100	Intro Phy
Jane Doe	E100	English I
John Public	C100	Intro Chem
John Public	P100	Intro Phy

If the student name "Jane Doe" changes, all occurrences of the name must be changed.

I1120

The following example shows a surrogate key being used instead of a natural key. Maintaining data is less complex than it is with natural keys and consequently less error-prone.

BP1257

Statement:

Place a **unique key constraint** on the **natural key** fields.

Rationale:

Surrogate keys make it easier to maintain data. However, a column or set of columns should still uniquely identify the row in the table. This column or set of columns is the "natural key" or "secondary key." This natural key should still be protected by the uniqueness constraint normally associated with a **primary key**.

Referenced By:

RDBMS Internals

Evaluation Criteria:

1) Test: [BP1257.1]

Is there a unique key index for all tables that includes a column or set of columns not including the primary key?

Procedure:

Look at the database metadata to ensure that each table has a unique key, and that the columns in the unique key are not also part of the primary key.

Example:

BP1375

Statement:

Use **asymmetric encryption** for **SOAP**-based **Web services**.

Rationale:

Most Web services exchange very few messages so the fact that asymmetric encryption is computationally intensive is a non-issue. Symmetric encryption is more efficient, but it is done by sharing a secret key outside the SOAP message communication which is less portable.

Referenced By:

XML Web Service Security
Design Tenet: Encryption and HAIPE
Design Tenet: Identity Management, Authentication, and Privileges

BP1392

Statement:

Register services in accordance with a documented service registration plan.

Rationale:

Program information services are provided via a shared space for use by consumers. In order to locate these services and access the corresponding information provided, the services should be registered in the **service registry** per direction of the shared information space manager.

Referenced By:

Design Tenet: Be Responsive to User Needs
Design Tenet: Make Data Visible
Metadata Registry
Design Tenet: Make Data Accessible
Design Tenet: Make Data Interoperable
Design Tenet: Provide Data Management
Design Tenet: Make Data Understandable
Interoperability
Reusability

Evaluation Criteria:

1) Test: [BP1392.1]

Has the Program generated default service definitions and registered them in the DoD service registry?

Procedure:

Review that there is a service definition (URLs, WSDL entries, etc.) for each of the program information services and that they have been registered accordingly.

Example:

None

BP1400

Statement:

Programs will use authoritative **metadata** established by the Joint Mission Threads (JMTs) when available.

Rationale:

Referenced By:

Design Tenet: Joint Net-Centric Capabilities
Data Modeling

BP1404

Statement:

For DoD Programs requiring a **data model**, the **NATO** Generic Hub v.5 model (**LC2IEDM**) is an example of a successful **COI**-developed model.

Rationale:

The **Land C2 Information Exchange Data Model** (**LC2IEDM**), or Generic Hub (GH, now version 5) model has been under development in the **NATO** environment. This model is a rich Joint battlespace operational context model. Many NATO countries have developed prototypes. The U.S. Army has also been active in the Generic Hub efforts.

Referenced By:

Reading/Writing Objects within a DDS Domain
Metadata Registry

BP1594

Statement:

Examine the use of **Transmission Control Protocol** (TCP) extentions and other transport protocols that have been designed to mitigate risk for high bandwidth, high latency satellite communications.

Rationale:

TCP performance over satellite links is generally poor due to delays and blockages inherent to satellite links. TCP extensions (e.g., **IETF** RFC 1323) and other transport protocols that have been developed to mitigate this risk should be considered for high bandwidth, high latency satellite communications.

Referenced By:

Mobile Nodes
Design Tenet: Transport Goal

Evaluation Criteria:

1) Test: [BP1594.1]

If the system is involved in high bandwidth, high latency satellite communications, does the Node design address TCP performance?

Procedure:

Determine if parts of the system involve high bandwidth, high latency satellite communications and if so, look for a TCP extension.

Example:

None.

BP1614

Statement:

Prepare a **Node** for the possibility of becoming a new **Component** service within another Node.

Rationale:

While the complexities of nested Nodes are currently not addressed within NESI Part 4, nested Nodes are a possibility; thus, Nodes should be prepared to interact in such an environment. Following the guidance for Nodes in Part 4 should be sufficient to prepare the Node for such interactions by encouraging the proper definition of key interfaces and capabilities and creating a distinction between Nodal infrastructure and Component capabilities. These distinctions would allow a Node, for example, to supplant it's own infrastructure with those of it's new parent Node (either directly or via proxies).

Note: *The purpose of this practice is not necessarily to encourage nested Nodes, but to ensure that Nodes apply appropriate open* ***modular designs*** *both externally and internally to ensure greater interoperability in a variety of environments.*

Referenced By:

Web Client Platform
Design Tenet: Cross-Security-Domains Exchange
Cross-Domain Interoperation

Evaluation Criteria:

1) Test: [BP1614.1]

Does the Node use standardized interfaces to obtain the services of routine activities?

Procedure:

Look for alignment and adherence to guidance of NESI Part 4 and open systems approaches.

Example:

None.

BP1651

Statement:

Do not implement **server** side **CES** functionality for **Components**.

Rationale:

The burden of aligning to standard CES functionality and providing the functionality uniformly rests on the Node infrastructure, rather than the **Components** within the Node. This isolates the Components from the **CES** complexity and enhances portability and interoperability of the Components.

Referenced By:

Design Tenet: Network Connectivity
CES and Intermittent Availability

Evaluation Criteria:

1) Test: [BP1651.1]

Do any **Component** systems, applications or services implement any of the server side **CES** Global Information Grid (**GIG**) **Key Interface Profiles** (KIPs)?

Procedure:

Review the Component systems, applications or services code for implementations of the server side CES Global Information Grid (GIG) Key Interface Profiles (KIPs).

Example:

None.

BP1661

Statement:

Engage with the **Net-Centric Enterprise Services** (NCES) program office to explore approaches for mobile use of the **Core Enterprise Services** (CES) services in mobile Nodes that rely on **Transmission Control Protocol/ Internet Protocol** (TCP/IP) for inter-node communication.

Rationale:

Referenced By:

CES Definitions and Status
Design Tenet: Joint Net-Centric Capabilities

BP1663

Statement:

Design a **Domain Name System** (DNS) in coordination with the appropriate governing Internet Protocol Version 6 (IPv6) Transformation Office.

Rationale:

Referenced By:

Design Tenet: IPv6
Domain Name System (DNS)

BP1669

Statement:

Select **XML**-capable **trusted guards**.

Rationale:

As **XML** is a fundamental transfer format for data in interoperable net-centric environments, **trusted guards** should be capable of transferring XML data to facilitate cross-domain interoperability.

Referenced By:

Design Tenet: Cross-Security-Domains Exchange
Trusted Guards

BP1670

Statement:

Monitor Black Core implementation issues and prepare a plan for local implementation in coordination with system programs fielded within the Node.

Rationale:

Referenced By:

Design Tenet: Concurrent Transport of Information Flows
Black Core

BP1671

Statement:

Consider Black Core transition whenever there is a significant Node network design or configuration decision to make in an effort to avoid costly downstream changes caused by Black Core transition.

Rationale:

Referenced By:

Black Core
Design Tenet: Concurrent Transport of Information Flows

BP1672

Statement:

Be prepared to integrate fully with the **Information Assurance** (IA) infrastructure.

Rationale:

Referenced By:

Design Tenet: Net-Centric IA Posture and Continuity of Operations
Web Client Platform

BP1681

Statement:

Make **Component** services metrics visible and accessible as part of the service registration and updated periodically.

Rationale:

Metrics are normally also needed to ensure performance is provided according to more traditional **Service Level Agreements** (SLAs) and for operations management.

Referenced By:

Instrumentation for Metrics
Design Tenet: Joint Net-Centric Capabilities

BP1686

Statement:

Align Node interfaces to **Components** for directory services with the guidance being provided by the Joint Enterprise Directory Services Working Group (JEDIWG) and sub-working groups, including such guidance as naming conventions, federation, and synchronization.

Rationale:

Referenced By:

Design Tenet: Joint Net-Centric Capabilities
Directory Services

BP1689

Statement:

Use the **Service Discovery** (SD) pilot program to practice and exercise the mechanics of service discovery and late binding.

Rationale:

The pilot program provides an opportunity to practice and exercise the mechanics of **Service Discovery** (SD) and late binding.

Referenced By:

Service Discovery
Design Tenet: Service-Oriented Architecture (SOA)

BP1691

Statement:

Use **Node** implemented **Service Discovery** (**SD**) to meet compartmentalization needs.

Rationale:

For pilot implementations that are not reachable, such as might be the case in a higher classified environment, the Nodes should coordinate among themselves and DISA to provide pilot and full service implementations that are reachable.

Referenced By:

Design Tenet: Cross-Security-Domains Exchange
Cross-Domain Interoperation
Service Discovery

BP1698

Statement:

Plan for the event that **Component** services within a **Node** cannot be invoked across security domains.

Rationale:

Until such approaches are prototyped and explored more fully, Nodes should anticipate that services will not be capable of cross-domain invocation.

Referenced By:

Cross-Domain Interoperation
Design Tenet: Cross-Security-Domains Exchange

BP1701

Statement:

Configure **Components** for **Information Assurance** (IA) in accordance with the Network **Security Technical Implementation Guide** (STIG).

Rationale:

Referenced By:

Design Tenet: Net-Centric IA Posture and Continuity of Operations
Network Information Assurance

BP1705

Statement:

Design **DNS** infrastructure in accordance with appropriate governing **IPv6** Transition Office requirements.

Rationale:

Referenced By:

IPv4 to IPv6 Transition
Domain Name System (DNS)
Design Tenet: IPv6

BP1712

Statement:

Register developed mappings in the **DoD Metadata Registry**.

Rationale:

Referenced By:

Mediation Services
Design Tenet: Joint Technical Architecture [now DISR]

BP1715

Statement:

Design SCA log services according to the OMG Lightweight Log Service Specification.

Rationale:

One component of the SCA framework is a central logging facility, enabling the asynchronous collection of informational messages from any component connected to the framework; and the controlled read access to this information. The Lightweight Logging Service is a free-standing, self-contained service which is not connected to an event channel or similar infrastructure. Using a standard log service specification between SCA implementations can improve interoperability and portability.

Referenced By:

Software Communication Architecture
Design Tenet: RF Acquisition

Evaluation Criteria:

1) Test: [BP1715.1]

Is the logging service designed according to the OMG Lightweight Log Service Specification? Is the logging service designed according to the OMG Lightweight Log Service Specification?

Procedure:

Check the log service provider's documentation for compliance with the OMG Lightweight Log Service Specification.

Example:

BP1732

Statement:

Follow the Upper Camel Case (UCC) naming convention for XML Type names.

Rationale:

The predominate style used by most programs or projects is to use the Upper Camel Case (UCC) for type names. Type names should be easy to differentiate from namespace prefixes and from attributes. Since the namespace prefix and the type name are separated by a non-whites character (i.e., the colon, :), it is easier to identify the type name from the namespace name if the type name follows the UCC.

Referenced By:

Defining XML Schemas
Defining XML Types

Evaluation Criteria:

1) Test: [BP1732.1]

Do type names follow the Upper Camel Case (UCC) naming convention?

Procedure:

Examine the schema definition and verify that the type names follow the Upper Camel Case (UCC) name convention.

Example:

```
<xsd:complexType
   name="MyType"
   . . .
</ xsd:coplexType>
```

BP1733

Statement:

Follow the Upper Camel Case (UCC) naming convention for **XML element** names.

Rationale:

The predominate style used by most programs or projects is to use the Upper Camel Case (UCC) for **XML element** names. Element names should be easily differentiable from namespace prefixes and from attributes. Since the namespace prefix and the element name are separated by a non-whites character (i.e., the colon, :), it is easier to identify the element name from the namespace name if the element name follows the UCC.

Referenced By:

Defining XML Schemas

Evaluation Criteria:

1) Test: [BP1733.1]

Do element names follow the Upper Camel Case (UCC) naming convention?

Procedure:

Examine the schema definition and verify that the element names follow the Upper Camel Case (UCC) name convention.

Example:

BP1734

Statement:

Follow the Lower Camel Case (LCC) naming convention for **XML attributes**.

Rationale:

The predominate style used by most programs or projects is to use the Lower Camel Case (LCC) for **XML attribute** names. Attributes are part of an attribute list which is a set of name="value" expressions separated by whitespace. Therefore, it is easy to find the beginning of the attribute name.

Referenced By:

Defining XML Schemas

Evaluation Criteria:

1) Test: [BP1734.1]

Do type names follow the Upper Camel Case (UCC) naming convention?

Procedure:

Examine the schema definition and verify that the type names follow the Upper Camel Case (UCC) name convention.

Example:

BP1790

Statement:

Stipulate that the Offeror is to describe how the proposed technical solution reuses services from other systems or demonstrates composeability and extensibility by building from existing reusable components and/or services.

Rationale:

Reuse of existing components and services leads to reduced costs and promotes modularity and composability. Reusable artifacts are common in large distributed networks. Future systems will be required to demonstrate composing new solutions from reusable components and services.

Referenced By:

Section L: Instructions, Conditions, and Notices to Offerors
Design Tenet: Layering and Modularity

Evaluation Criteria:

1) Test: [BP1790.1]

Does the Offeror demonstrate reuse of existing components or services?

Procedure:

Identify in the proposal the components or services identified as being reused.

Example:

None.

BP1824

Statement:

Use the **`USER_DATA`** **Quality of Service** (QoS) to communicate metadata on the **DomainParticipant** that may be used to authenticate the application trying to join the Data **Distribution Service** (DDS) **Domain**.

Rationale:

In many cases the application needs to send additional information that describes the **`DomainParticipant`** to other participants in the DDS Domain. This information can be used to authenticate the participant or to meet any other application-specific need.

The **`USER_DATA`** QoS on the **`DomainParticipant`** allows the application to store un-interpreted bytes that will be propagated via the DDS built-in discovery mechanism and will be accessible to the other **`DomainParticipants`** on the system.

Referenced By:

DDS Quality of Service

Evaluation Criteria:

1) Test: [BP1824.1]

Is the **`USER_DATA`** QoS set on the **`DomainParticipant`**?

Procedure:

Check the creation of the **`DomainParticipant`** and determine whether the **`USER_DATA`** QoS is used. Ensure that the application does not use another non-standard way to accomplish the same function.

Example:

None.

BP1826

Statement:

Use the **`USER_DATA`** **Quality of Service** (QoS) on the **DataWriters** and **DataReaders** to communicate metadata that may provide application-specific information of the entity writing/reading data in a **Data Distribution Service** (DDS) **Domain**.

Rationale:

In many cases the application needs to send additional information that describes the **`DataWriter`** or the DataReader to other entities in the DDS Domain. This information can be used to authenticate the **`DataWriter`**/ Reader or to meet any other application-specific need.

The **`USER_`**DATA QoS on the **`DataWriter`** and the **`DataReader`** allows the application to store un-interpreted bytes that will be propagated via DDS's built-in discovery mechanism and will be accessible to the other **`DataWriters`** and **`DataReaders`** on the system.

Referenced By:

DDS Quality of Service

Evaluation Criteria:

1) Test: [BP1826.1]

Is the **`USER_DATA`** QoS set on the **`DataWriter`** and **`DataReader`**?

Procedure:

Check the creation of the **`DataWriter`** and **`DataReader`** and determine whether the **`USER_DATA`** QoS is used. Ensure that the application does not use another non-standard way to accomplish the same function.

Example:

None.

BP1829

Statement:

Use the **Data Distribution Service** (DDS) **`OWNERSHIP`** **Quality of Service** (QoS) kind set to **`EXCLUSIVE`** when multiple **DataWriters** cannot write each unique data-object within a DDS **Topic** simultaneously.

Rationale:

DDS easily supports multiple **publishers** adding data to the same topic without impacting the **subscribers**. Using the DDS **`OWNERSHIP`** QoS kind set to **`EXCLUSIVE`** places the entire burden off supporting the multiple publishers on the DDS implementation rather than the publisher or subscriber code. This results in an increase of modularity, portability and the maintainability.

Referenced By:

Design Tenet: Layering and Modularity
DDS Quality of Service

BP1830

Statement:

Use the **Data Distribution Service** (DDS) Content Profile to tailor subscription message data.

Rationale:

The DDS Content Profile allows for the **subscribers** to select and refine the data that is retrieved from a **Topic**. This tailoring code is part of the DDS infrastructure and is well tested and reliable. Not using the DDS Content Profile and using code within the subscriber increases the complexity of the subscriber and causes tight coupling between the subscriber code and the Topic.

Referenced By:

Design Tenet: Network Connectivity
Decoupling Using DDS and Publish-Subscribe

BP1832

Statement:

Handle all **Data Distribution Service** (DDS) **Data Local Reconstruction Layer** (DLRL) Exceptions.

Rationale:

The DLRL API may raise Exceptions under certain conditions. The following is an extensive list of all possible Exceptions and the conditions in which they will be raised:

BP1833

Statement:

Use the **Data Distribution Service** (DDS) Object Model Profile for accessing message data as objects.

Rationale:

The DDS **Data Local Reconstruction Layer** (DLRL) is intended to provide an abstraction layer between the actual underlying data and the higher level object level concepts used in applications. The Object Model Profile defines how applications interact with the abstract object layer. Applications that are bound directly to the actual underlying data are tightly coupled to the layer and are subject to its evolutionary changes.

Note: *DLRL, a recent addition to the DDS specification is particularly rich; implementations using this upper level profile of the specification are still emerging.*

Referenced By:

DDS Data Local Reconstruction Layer (DLRL)

BP1836

Statement:

Obtain consensus on the migration plan from all key **stakeholders**.

Rationale:

The stakeholders present varying viewpoints about issues associated with the migration plan. Obtaining consensus from key stakeholders on the migration plan can prevent critical miscommunication and support the management of expectations.

Referenced By:

Finalize Migration Plan
Migration Planning Process

Evaluation Criteria:

1) Test: [BP1836.1]

Does the migration plan identify key stakeholders?

Procedure:

Examine the migration plan and verify that it identifies key stakeholders.

Example:

None.

2) Test: [BP1836.2]

Does the migration plan reflect key stakeholders' involvement and input?

Procedure:

Examine and analyze the migration plan to confirm that it reflects key stakeholders' involvement and input.

Example:

None.

BP1837

Statement:

Update the **net-centric** and SOA migration plan in an iterative manner as the program gains migration experience and conditions change.

Rationale:

Most large-scale net-centric and SOA migrations are expected to be lengthy and subject to many influencing and changing factors. As a result, they should be implemented in phases. Small-scale migrations may be able to execute the bulk of the migration in a single increment, but the migration plan should still be revisited for potential updates over time. Specifically, use the same methodology for creating updates to the plan as for creating the initial baseline version.

Referenced By:

Migration Planning Process
Design Tenet: Joint Net-Centric Capabilities
Finalize Migration Plan

Evaluation Criteria:

1) Test: [BP1837.1]

Does the migration plan track its currency date and any updates?

Procedure:

Examine the migration plan for a currency date and update tracking.

Example:

None.

BP1840

Statement:

Identify opportunities to apply the principles of net-centricity and SOA throughout the course of the program.

Rationale:

All of the program's modernization activities have the potential to include opportunities to migrate to net-centricity and SOA. Even requirements that on the surface appear to not relate to net-centricity or SOA may contain a net-centric or SOA aspect. Coordinate with both user and developer personnel to identify these opportunities and the associated risks. Be careful to not overstate the requirements.

Referenced By:

Design Tenet: Joint Net-Centric Capabilities
Assess As-Is Requirements

Evaluation Criteria:

1) Test: [BP1840.1]

Does the program's migration plan describe an approach for identifying opportunities to apply net-centric and SOA principles throughout the course of the program?

Procedure:

Verify that the migration planning documentation contains a description of an approach for identifying net-centric and SOA migration opportunities.

Example:

None.

2) Test: [BP1840.2]

Does the program's migration plan contain an analysis of opportunities to apply net-centric and SOA principles throughout the course of the program?

Procedure:

Review the program's migration planning documentation and verify that it contains an analysis of opportunities of opportunities to apply net-centric and SOA principles throughout the course of the program.

Example:

None.

BP1842

Statement:

Formally document the migration rationale to support the migration to net-centricity and SOA.

Rationale:

A clearly documented rationale presents the business case for the migration to all stakeholders.

Referenced By:

Develop Migration Rationale Statement

Evaluation Criteria:

1) Test: [BP1842.1]

Does the program have a migration rationale statement to support the migration to net-centricity and SOA?

Procedure:

Review migration planning documents to verify they include a migration rationale statement.

Example:

None.

2) Test: [BP1842.2]

Does the Migration Plan include a formally documented migration rationale?

Procedure:

Review the Migration Plan to verify it includes a migration rationale.

Example:

None.

BP1843

Statement:

Obtain consensus among all key stakeholders on the rationale for the migration to net-centricity and SOA.

Rationale:

The stakeholders present varying viewpoints about issues associated with the migration. Involving them early on in the migration planning process provides key input and potential advocacy.

Referenced By:

Develop Migration Rationale Statement

Evaluation Criteria:

1) Test: [BP1843.1]

Does the Migration Rationale statement explicitly demonstrate the consensus on the rationale for the migration to net-centricity and SOA among all of the key stakeholders?

Procedure:

Review the Migration Rationale statement and verify that it demonstrates all key stakeholders consensus.

Example:

None.

BP1844

Statement:

Develop a vision statement for the migration to net-centricity and SOA.

Rationale:

A vision statement provides strategic direction for the migration. It describes the high-level, time-indeterminate state of the target of the migration. The vision statement will be documented in the migration plan.

The vision for the program indicates the desired long-term direction for the system. It offers a view of its evolution and, potentially, eventual replacement. The vision for the program shows the scope of the system within its larger context (the enterprise); thus, the vision for the program should be consistent with the higher headquarters vision statements. Similarly, the vision for the migration to net-centricity and SOA should be consistent with the vision for the program.

Referenced By:

Develop Alternative Target Architectures

Evaluation Criteria:

1) Test: [BP1844.1]

Does the migration plan contain a vision statement for the migration?

Procedure:

Review the migration plan and verify that it contains a migration vision statement.

Example:

None.

BP1845

Statement:

Consider key enterprise-level concerns when planning and executing a migration to net-centricity and SOA.

Rationale:

The complexity of migration planning and execution requires careful consideration of numerous factors. Early and deliberate consideration of these factors is required to successfully achieve both program and enterprise-level objectives associated with the migration.

Referenced By:

Develop Implementation Plans
Design Tenet: Network Connectivity
Critical Migration Concerns

Evaluation Criteria:

1) Test: [BP1845.1]

Does the implementation plan for net-centricity and SOA migration contain considerations for key enterprise-level concerns?

Procedure:

Review the migration plan tasks and verify that they address critical migration concerns.

Example:

None.

BP1846

Statement:

Involve key stakeholders in the development of the implementation plan increments.

Rationale:

The stakeholders present varying viewpoints about issues associated with the migration. Involving them in the migration planning process provides key input and potential advocacy.

Referenced By:

Develop Implementation Plans

Evaluation Criteria:

1) Test: [BP1846.1]

Does the implementation plan for net-centricity and SOA migration contain considerations of key stakeholders?

Procedure:

Review the migration plan tasks and verify that they address key stakeholders' concerns.

Example:

None.

BP1863

Statement:

Make shareable data assets visible, even if they are not accessible.

Rationale:

Making data visible using a consistent, standardized metadata specification within a Net-Centric Environment (NCE) facilitates a federated cross-organizational discovery capability [R1172] . A common specification for the description of information allows for a comprehensive capability that can locate all information across the NCE regardless of format, type, location, or classification, dependent on user authorization. The **DoD Metadata Specification** (**DDMS**) was developed to support Enterprise-wide data discovery by providing a common set of descriptive metadata elements. Discovery metadata must conform to the DDMS in accordance with DoD Directive (DoDD) 8320.2 [R1217] . Information owners tag information with DDMS-compliant metadata to ensure discoverability of information in the NCE.
The extensible nature of the DDMS supports domain-specific or **COI** discovery metadata requirements and extends the element categories identified in the DDMS Core Layer used to describe information. Use of the DDMS does not preclude use of other metadata processes or standards. For example, record-level database tagging and in-line document tagging are common practices to support various department objectives. These tagging initiatives should be enhanced to include the DDMS for enterprise discovery.

Referenced By:

Design Tenet: IPv6
Net-Centric Data Strategy (NCDS)
Design Tenet: Make Data Visible
Design Tenet: Open Architecture
Design Tenet: Service-Oriented Architecture (SOA)

Evaluation Criteria:

1) Test: [BP1863.1]

Does the system provide discovery metadata in accordance with the DoD Discovery Metadata Standard (DDMS) for all data posted to shared spaces?

Procedure:

Examine the DoD Metadata Registry for program/system.

Example:

Discoverable information has associated DDMS metadata that can be found in the DDMS).

BP1864

Statement:

Layer architectures to support clear boundaries between data management, presentation, and business logic functionality.

Rationale:

Multitier, or n-tier, architectures are types of client/server architectures that enable an application to be accessed and executed by one or more software agents or services on the network. An N-tier architecture should be composed of layers; **graphical user interface** (**GUI**), business logic, and data should enable developing and maintaining each tier separately as technologies change. Separation of each tier may be logical or physical. Regardless of the physical system design, the structure should include well-defined boundaries between the different tiers so that changes in the system are transparent to users.
For example, N-tier architectures may employ Web services as a means of separating the presentation layer from business logic and data layers. The presentation layer serves static content through **Web pages**. A business logic layer provides dynamic content using a **J2EE application server**. Finally, a database provides the underlying information that must be shared.

Referenced By:

Design Tenet: Packet Switched Infrastructure
Design Tenet: Scalability
Design Tenet: Open Architecture
Design Tenet: Transport Goal
Design Tenet: Accommodate Heterogeneity

Evaluation Criteria:

1) Test: [BP1864.1]

Does the architecture support clear boundaries between data, presentation, and business logic layers?

Procedure:

Examine the architecture for clear boundaries between data, presentation, and business logic layers.

Example:

The architecture uses Web Services to share information between the presentation and business logic layers.

BP1865

Statement:

Provide sufficient program, project, or initiative **metadata** descriptions and automated support to enable **mediation** and translation of the data between **interfaces**.

Rationale:

Information exchanges should support known and unanticipated users. The program or project should initiate sufficient metadata descriptions and provide automated support to enable mediation and translation of data between interfaces. All of the data that can and should be shared externally beyond the programmatic bounds of your program should be defined well enough in metadata descriptions and translation of the data between interfaces should be automated.

Referenced By:

Content Discovery Services
Net-Centric Data Strategy (NCDS)
Design Tenet: Provide Data Management
Design Tenet: Make Data Visible
Net-Centric Information Engineering
Metadata
Coordination of Node and Enterprise Services
Design Tenet: Make Data Interoperable

Evaluation Criteria:

1) Test: [BP1865.1]

Evaluation of interfaces and applicable mediation/translations to access that the program, project, or initiative has sufficient metadata descriptions and automated support to enable mediation and translation of the data between interfaces. Data is XML wrapped for exchange and configured to support standard transactions with headers, trailers and bodies.

Procedure:

Evaluate the degree to which data is XML wrapped for exchange and configured to support standard transactions with headers, trailers and bodies.

Evaluation of the DoD Metadata Registry entries to assess sufficient metadata descriptions and automated support the enables mediation and translation of the data between interfaces.

Example:

XML wrapped data are intend for exchange, that is configured in terms of standard transactions with headers, trailers and bodies.

BP1866

Statement:

Coordinate with end users to develop interoperable materiel in support of high-value mission capability.

Rationale:

System providers acquire the materiel portion of mission capabilities that include all aspects of DOTMLP-F. An assessment by the community regarding the value of information or services provides useful direction in support of managing a mission area's portfolio of services. User feedback mechanisms provide a means of capturing and reporting user satisfaction and give portfolio managers decision-making information to steer investments, developments, and improvements. As service consumers gain access to information more quickly in the operational environment, command structures will inevitably change the manner in which IT investments are made. Service and information providers in a mission area should work together to define the processes for using the user feedback for service and information improvements because these processes are specific to a portfolio of capabilities in the Enterprise.

Referenced By:

Design Tenet: Make Data Interoperable
Net-Centric Information Engineering
Design Tenet: Joint Net-Centric Capabilities

Evaluation Criteria:

1) Test: [BP1866.1]

Processes exist that allow a consumer to

1. request changes in the format (syntax or semantic) of the visible data asset;
2. report a problem with a data asset;
3. request additional data from the data provider

Procedure:

Evaluation of the process a consumer would follow to

1. request changes in the format (syntax or semantic) of the visible data asset;
2. report a problem with a data asset;
3. request additional data from the data provider.

Example:

An end-to-end output management strategy, across multiple business sites and/or the enterprise.

A distributed and extensible database which make information accessible to authorized users across the enterprise.

BP1867

Statement:

Use metrics to track responsiveness to user information sharing needs.

Rationale:

Information sharing metrics are defined to measure and track implementation of the net-centric approaches. Measurement techniques should be developed to ensure that metrics are captured in a useful and consistent manner. Metrics should be tagged with **DDMS**-compliant metadata and provided to the NCE to promote awareness of data management successes and areas requiring improvement.

Referenced By:

Design Tenet: Be Responsive to User Needs
Instrumentation for Metrics

Evaluation Criteria:

1) Test: [BP1867.1]

Does the program, project or initiative have metrics for determining responsiveness to user needs?

Procedure:

Evaluate the metrics being used to determine responsiveness to user data needs. If YES, describe; If NO, explain and identify a time frame for when the program, project, or initiative will have metrics for determining responsiveness to user needs; or specify NOT APPLICABLE and explain.

Example:

Examples of data metrics include percentage of Web-enabled components, progress toward service-enabling identified key functional components, and percentage of tagged community data.

BP1868

Statement:

Incorporate mechanisms to enhance the survivability, resiliency, redundancy, and reliability of Computing Infrastructure (CI).

Rationale:

Computing Infrastructure (CI) must be survivable, resilient, redundant, and reliable in the presence of attacks, failures, accidents, and natural or man-made disasters. A robust CI must incorporate survivability, resiliency, redundancy, and reliability to ensure operational availability in support of information sharing in DoD, as well as externally with federal agencies, state and local governments, allies, and coalition partners. In the context of the CI, the measure of reliability is included as a critical element in ensuring high mean time between failures (MTBF).

Survivable: Survivability ensures that CI systems, subsystems, equipment, processes, procedures, or CI-related doctrine, organization, training, materiel, leadership, personnel, facilities (DOTMLPF) continue to fulfill critical mission requirements in the presence of attacks, failures, accidents, and natural or man-made disasters.

Resilient: Incorporation of resiliency into CI ensures the ability to automatically recover from, or adjust to, attacks, failures, or accidents. Fault tolerance is a key example of resilience that measures the ability to respond gracefully to an unexpected CI system, subsystem, process, or procedure failure.

Redundant: Incorporation of automatic redundancy into the CI ensures that alternative devices are available to perform the required system functionality if a primary device fails. Redundancy also ensures that system data remains accessible and corruption free when CI components fail.

Reliable: Reliable OS platforms, other software infrastructure, and hardware components are critical to ensuring that operators can depend on their ability to support system functions and applications. Bandwidth conservation mechanisms minimize latency and jitter, as well as the instability that comes from running processors and networks with high loads. Processing efficiency mechanisms, such as efficient software implementation techniques, allow applications to meet performance and latency requirements. Typically, reliability is measured in mean time between user failures (MTBUF). MTBF of CI components is one factor affecting the overall system MTBF.

A Continuity of Operations Plan (COOP) and disaster recovery planning are also key to ensuring a robust CI. The DoD Dictionary of Military Terms defines COOP as "the degree or state of being continuous in the conduct of functions, tasks, or duties necessary to accomplish a military action or mission in carrying out the national military strategy." It includes the functions and duties of the commander, as well as the supporting functions and duties performed by the staff and others acting under the authority and direction of the commander.

Referenced By:

Design Tenet: Availability
Design Tenet: Enterprise Service Management

Evaluation Criteria:

1) Test: [BP1868.1]

Does the program or initiative have a Continuity of Operations Plan (COOP) plan?

Procedure:

Verify existence of COOP.

Example:

Continuity of Operations Plans and Disaster Recovery Plans that include preparatory measures, response actions, and restoration activities planned or taken to ensure continuation of critical functions to maintain effectiveness, readiness, and survivability.

Technologies that allow, self-correcting mechanisms to be implemented (e.g., automatic recovery without manual intervention).

Clustering of servers, incorporation of relative addressing schemata (e.g., **DNS**), site mirroring, and provisioning of geographically distributed CI functionality are examples of fail-over implementations.

BP1870

Statement:

Conform to DoD-specified data publication methods that are consistent with **Global Information Grid** (**GIG**) enterprise and user technologies per DoD Directive 8101.1. [R1166]

Rationale:

Referenced By:

Design Tenet: IPv6
Design Tenet: Accommodate Heterogeneity

BP1874

Statement:

Develop methods to forward IP datagrams from external networks.

Rationale:

A system should have the ability to act as a transit network for IP datagrams where the origin and/or destination are external to the system.

Referenced By:

Design Tenet: Make Data Accessible
Design Tenet: Packet Switched Infrastructure

Evaluation Criteria:

1) Test: [BP1874.1]

Does the system have method(s) to accept IP datagrams from external networks that are destined for hosts within the system?

Procedure:

Identify method(s) used to accept IP datagrams from external networks destined for hosts within the system.

Example:

None.

2) Test: [BP1874.2]

Is the system able to act as a transit network of IP datagrams with an origin and destination that are external to the system?

Procedure:

Verify that the system can act as a transit network for IP datagrams with an origin and destination that are external to the system.

Example:

None.

BP1875

Statement:

Describe the process and protocols used to provide concurrent traffic from multiple security domains on a single **IP** internetwork.

Rationale:

Transport service users should implement interfaces to (or transition to) a transport infrastructure supporting fully converged IP traffic (voice, video, data, and imagery) using DoD-adopted standards (see **DISR** for appropriate standards). Transport service providers should implement converged nets as a single IP internetwork. DoD requires multiple security domains to conduct network-centric warfare.

Referenced By:

Design Tenet: Concurrent Transport of Information Flows
Design Tenet: Joint Technical Architecture [now DISR]
Design Tenet: Transport Goal

Evaluation Criteria:

1) Test: [BP1875.1]

What processes and protocols are used to provide convergence of traffic (voice, video and data) from multiple security domains on a single IP internetwork?

Procedure:

Describe the process (and protocols) used to provide convergence of traffic (voice, video and data from multiple security domains on a single IP internetwork. Verify that DoD standards and products to support traffic convergence are utilized.

Example:

NSA-approved multi-level security guard.

BP1876

Statement:

Provide a priority-based differentiated management of **quality-of-service** for traffic based on class of user, application, or mission.

Rationale:

The GIG and its components must support both QoS and CoS in accordance with the DoD QoS/CoS Roadmap and policies. The primary QoS factors that affect end-user experience include availability, throughput, delay/latency, jitter (variation in delay with time), and bit/packet loss. In addition, all GIG networks should be designed with the ability to support end-to-end treatment of multiple distinct classes of service prioritization levels. These prioritization levels require that higher-precedence data flows will be transmitted through the networks with their required QoS with greater assurance than are lower-precedence data flows. Prioritization must enforce transmission of higher-precedence data in the network, at best, concurrently with or, at worst, to the detriment of lower-precedence data flows. In the best case, sufficient resources exist to transmit data of different priorities with their required quality. Otherwise, higher-priority data must be transmitted at the expense of lower-precedence data, possibly degrading or even preempting the lower-priority data.This capability, referred to as Class of Service (CoS) support, corresponds approximately to the notion of Multi-Level Priority and Preemption (MLPP).

Referenced By:

Design Tenet: Transport Goal
Design Tenet: Differentiated Management of Quality-of-Service
Design Tenet: Packet Switched Infrastructure
Design Tenet: Layering and Modularity

Evaluation Criteria:

1) Test: [BP1876.1]

Does the program, project, or initiative support a priority-based differentiated management QoS?

Procedure:

Describe the approach used to provide a priority-based differentiated management of quality-of-service.

Example:

Some applications in the GIG require firm service guarantees, while others operate correctly if they receive services that are differentiated with respect to one or more performance characteristics.
Differentiated Services or DiffServ aggregates flows into coarse classes and then treats the packets in these classes differentially. Due to this aggregation, and the resulting absence of a need to consider individual flows beyond the edges of an internet, DiffServ exhibits good scaling properties. However, in the absence of additional mechanisms, DiffServ provides only preferential, differentiated levels of service and not guarantees.

BP1877

Statement:

Align end-to-end interoperable management of **QoS** with external networks.

Rationale:

QoS/CoS Working Group is investigating complete end-to-end QoS frameworks providing both differentiated and guaranteed QoS. They are developing a DoD roadmap and baseline architecture strawman. The architecture needs to define transport user and transport provider functions, such as where packets are labeled (application or router with Service Level Agreement).

Referenced By:

Design Tenet: Packet Switched Infrastructure
Design Tenet: Differentiated Management of Quality-of-Service
Design Tenet: Transport Goal

Evaluation Criteria:

1) Test: [BP1877.1]

Does the program, project, or initiative support end-to-end interoperable management of QoS with external networks?

Procedure:

Describe the approach used to provide a priority-based differentiated management of quality-of-service across external networks.

Example:

Complete end-to-end QoS frameworks providing both differentiated and guaranteed QoS.

BP1878

Statement:

Quantitative measures of QoS requirements should be supportable.

Rationale:

All GIG networks should be provisioned according to SLAs to provide QoS that meets or exceeds that required by networked applications for the transport of voice, data, video, imagery, and any other demands. The primary QoS factors that affect end-user experience include availability, throughput, delay/latency, jitter (variation in delay with time), and bit/packet loss.

Referenced By:

Design Tenet: Packet Switched Infrastructure
Design Tenet: Differentiated Management of Quality-of-Service
Design Tenet: Transport Goal

Evaluation Criteria:

1) Test: [BP1878.1]

What measures of quantitative QoS requirements are supportable, for example jitter, latency, throughput, packet loss, and others, under specific workloads?

Procedure:

Identify and describe all the QoS measurement criteria that the program, project or initiative will measure.

Example:

Jitter, latency, throughput, packet loss, etc.

BP1879

Statement:

The program, project or initiative should align with the DoD Qos/CoS Working Group Roadmap.

Rationale:

Various approaches are being explored, with none yet adopted. DoD QoS/CoS Working Group is investigating complete end-to-end QoS frameworks providing both differentiated and guaranteed QoS. They are developing a DoD roadmap and baseline architecture strawman. The architecture needs to define transport user and transport provider functions, such as where packets are labeled (application or router with Service Level Agreement).

Referenced By:

Design Tenet: Packet Switched Infrastructure
Design Tenet: Differentiated Management of Quality-of-Service
Design Tenet: Transport Goal
Design Tenet: Concurrent Transport of Information Flows

Evaluation Criteria:

1) Test: [BP1879.1]

Is the program, project, or initiative aligned with the DoD QoS/CoS Working Group roadmap?

Procedure:

Describe your program's alignment with the DoD QoS/CoS working group roadmap.

Example:

None.

BP1880

Statement:

Justify, document, and obtain a waiver for all radio terminal acquisitions that are not JTRS/SCA compliant.

Rationale:

Tactical communications programs should focus on attaining the end objective of providing a family of software-programmable radios that will greatly enhance warfighters' wireless communication capabilities, while decreasing cost of ownership for infrastructure. The Joint Tactical Radio System (JTRS) will provide critical communications capabilities for the tactical wireless tails of the GIG. JTRS and its software communications architecture (SCA) continue to evolve and have become a cornerstone of the provision of future net-centric capabilities.

Referenced By:

Design Tenet: Joint Net-Centric Capabilities
Design Tenet: Concurrent Transport of Information Flows
Software Communication Architecture
Design Tenet: Employment of Wireless Technologies

Evaluation Criteria:

1) Test: [BP1880.1]

Are all of the program's, project's, or initiative's radio acquisitions JTRS/SCA compliant?

Procedure:

Describe all radio acquisitions that are not JTRS/SCA compliant.

Example:

None.

Glossary

.NET		To address the confusing maze of computer languages, libraries, tools, and toolkits that were necessary for creating multi-tier applications, Microsoft developed the .NET Framework and integrated it into Microsoft Windows as a component. It supports building and running multi-tier and service-oriented architectures, including Web services and client and server applications. It simplifies the process of designing, developing, and testing software, allowing individual developers to focus on core, application-specific code.
Active Server Page	ASP	A script that is executed by Microsoft Internet Information Services. The output is returned to the user as **HTML**. Typically, an ASP script generates a customized Web page on the fly before sending it to the user. ASPs are specific to Microsoft, only run on **IIS** or **PWS**, can contain HTML, **JScript**, and **VBScript**, and can access **COM** components.
ActiveX		An ActiveX control is similar to a Java **applet**. However, ActiveX controls have full access to the Windows OS. This gives them much more power than Java applets, plus a risk that the applet may damage software or data on your machine. To control this risk, Microsoft developed a registration system so that browsers can identify and authenticate an ActiveX control before downloading it. Another difference between Java applets and ActiveX controls is that Java applets can be written to run on all platforms, whereas ActiveX controls are currently limited to Windows environments.
Adapter		An intermediary that translates between incompatible components interfaces, allowing them to communicate.
All Views	AV	The DoDAF All-Views (AV) products provide information pertinent to the entire architecture but do not represent a distinct view of the architecture. AV products set the scope and context of the architecture. The scope includes the subject area and timeframe for the architecture. The setting in which the architecture exists comprises the interrelated conditions that compose the context for the architecture. These conditions include doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations; scenarios; and environmental conditions. (Source: *DoDAF* v1.5 Volume 1: Definintions and Guidelines, 23 April 2007)
American National Standards Institute	ANSI	Administrator and coordinator of the United States private-sector voluntary standardization system. ANSI facilitates the development of American National Standards (ANS) by accrediting the procedures of standards-developing organizations. The Institute remains a private, nonprofit membership organization supported by a diverse constituency

		of private and public sector organizations. (Source: http://web.ansi.org/)
Applet		A J2EE component that typically executes in a Web browser. Applets can also execute in a variety of other applications or devices that support the applet programming model. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Application		Provides the resources necessary to provision, operate and maintain **Net-Centric Enterprise Services** (NCES) capabilities.
Application Programming Interface	API	A special type of interface that specifies the calling conventions with which one component may access the resources and services provided by another component. APIs are defined by sets of procedures or function-invocation specifications. An API is a special case of an interface.
Application Server		A platform for developing and deploying multi-tier distributed enterprise applications.
Assistant Secretary of Defense for Networks and Information Integration	ASD (NII)	(Source: http://www.dod.mil/nii/)
Asymmetric Key Cryptography		Synonym for **Public Key Cryptography**.
Authorization		The process by which access to a method or resource is determined. Authorization depends on the determination of whether the principal associated with a request through authentication is in a given security role. A security role is a logical grouping of users defined by the person who assembles the application. A deployer maps security roles to security identities. Security identities may be principals or groups in the operational environment. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Basic Object Adapter	BOA	The Basic Object Adapter was an early (v1) CORBA component; see the **Portable Object Adapter** (**POA**).
Business Logic		The code that implements the functionality of an application. In the Enterprise JavaBeans architecture, this logic is implemented by the methods of an enterprise bean. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Capability Development Document	CDD	Provides operational performance attributes, including supportability, for the acquisition community to design the proposed system. Includes key performance parameters (KPP) and other parameters that guide the development, demonstration, and testing of the current increment. Outlines the overall strategy for developing full capability. (Source: http://www.dau.mil/pubs/glossary/12th_Glossary_2005.pdf)
Capability Production Document	CPD	Addresses the production attributes and quantities specific to a single increment of an acquisition program.

		Supersedes threshold and objective performance values of the CDD. (Source: http://www.dau.mil/pubs/glossary/12th_Glossary_2005.pdf)
Cascading Style Sheet	CSS	Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents. (Source: http://www.w3.org/Style/CSS/)
Certificate	CERT	A certificate which uses a digital signature to bind together a public key with an identity information such as the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual. (Source: http://en.wikipedia.org/wiki/Certificate_%28cryptography%29)
Certificate Authority	CA	A trusted organization which issues digital public key certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many public key infrastructure (PKI) schemes. (Source: http://en.wikipedia.org/wiki/Certificate_authority)
Certificate Revocation List	CRL	A list of certificates (more accurately, their serial numbers) which have been revoked, are no longer valid, and should not be relied upon by any system user. (Source: http://en.wikipedia.org/wiki/Certificate_Revocation_List)
Check Constraint		A constraint based on a user-defined condition - generally documented in a database domain - that has to evaluate to true for the contents of a data base column to be valid.
Client		A system entity that accesses a Web service. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Client-Certificate Authentication		An authentication mechanism that uses HTTP over SSL, in which the server and (optionally) the client authenticate each other with a public key certificate that conforms to a standard that is defined by X.509 Public Key Infrastructure. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Collaboration		Portal members can communicate synchronously through chat or messaging, or asynchronously through threaded discussion, blogs, and email digests (forums).
Command, Control, Communications, Computers, and Intelligence, Surveillance, and Reconnaissance	C4ISR	
Command and Control	C2	(DoD) The exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. Command and control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the

		accomplishment of the mission. (Source: http://www.dtic.mil/doctrine/jel/doddict/data/c/01093.htm)
Command and Control Information Exchange Data Model	C2IEDM	A data model that is managed by the Multilateral Interoperability Programme (MIP). It originated with experts from various NATO partners and from the Partnership-for-Peace nations. This data model is in the process of being submitted to OMG for consideration as the standard for information exchange. It falls under the shared operational picture exchange service. (Source: http://www.mip-site.org/MIP_DMWG.htm)
Commercial Off-The-Shelf	COTS	A term for systems that are manufactured commercially, and may be tailored for specific uses. (Source: http://en.wikipedia.org/wiki/Commercial_off-the-shelf)
Common Access Card	CAC	A DoD-wide smart card used as the identification card for active duty Uniformed Services personnel (to include the Selected Reserve), DoD civilian employees, eligible contractor personnel, and eligible foreign nationals; the primary platform for the Public Key Infrastructure (PKI) authentication token used to access DoD computer networks and systems in the unclassified environment and, where authorized by governing security directives, the classified environment; and the principal card enabling physical access to buildings, facilities, installations, and controlled spaces as described in DoD Directive 8190.3, "Smart Card Technology," 31 August 2002. (Source: DoD Instruction 8520.2, 1 April 2004, [R1206] Enclosure (2) Definitions, page 13)
Common Gateway Interface Script	CGI Script	CGI is a standard for interfacing external applications with information servers, such as HTTP or Web servers. A plain HTML document that the Web daemon retrieves is static, which means it exists in a constant state: a text file that doesn't change. A CGI program, on the other hand, is executed in real time, so it can output dynamic information.
Common Language Runtime	CLR	CLR, at the very core of the **.NET** Framework, encapsulates all the services used from the operating system by compilers of higher level languages such as Visual Basic .NET, Visual C++ .NET, Visual J# .NET and Visual C# .NET. The higher level languages ultimately are translated into native code that directly accesses the CLR.
Common Object Request Broker Architecture	CORBA	CORBA "wraps" code written in another language into a bundle containing additional information on the capabilities of the code inside, and explaining how to call it. The resulting wrapped objects can then be called from other programs (or CORBA objects) over the network. The CORBA specification defines APIs, communication protocol, and object/service information models to enable heterogeneous applications written in various languages running on various platforms to interoperate. (Source: http://en.wikipedia.org/wiki/CORBA)
Community of Interest	COI	A COI is a collaborative group of users that must exchange information in pursuit of its shared goals, interests, missions, or business processes nd therefore must have shared vocabulary for the information it exchanges. (Source: DoDD

		8320.02, 2 December 2004, *Data Sharing in a Net-Centric Department of Defense*)
Community of Interest Service		A service that may be offered to the enterprise, but is owned and operated by a **Community of Interest** to provide or support a well-defined set of mission functions and associated information.
Compiler		A computer program that translates programs expressed in a high-order language into their machine language equivalent. (Source: IEEE Std 610.12-1990)
Complex Data		Complex data can be represented in a complex data structure or can be mapped into a relational or flat structure with additional metadata provided to represent the complex relationships.
Component		One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components. Note the terms ***module***, ***component***, and ***unit*** are often used interchangeably or defined to be sub-elements of one another in different ways depending on the context. The relationship of these terms is not yet standardized. (Source: IEEE Std 610.12-1990) ***Note:*** *See **system component** and **software component**.*
Component Object Model	COM	A Microsoft software architecture for building component-based applications. COM objects are discrete components, each with a unique identity, which expose interfaces that allow applications and other components to access their features. COM objects are more versatile than Win32 DLLs because they are completely language-independent, have built-in inter-process communications capability, and easily fit into an object-oriented program design. COM was first released in 1993 with OLE2, largely to replace the inter-process communication mechanism DDE used by the initial release of OLE. **ActiveX** is based on COM.

Conceptual Model		Captures the concepts of the relational database and can help enforce the first three normalization rules.
Condition		A variable of the operational environment or situation in which a unit, system, or individual is expected to operate that may affect performance. A **DDS** Condition is attached to a **WaitSet** and indicates which condition the application is waiting for asynchronously: **`StatusCondition`**, **`ReadCondition`** or **`QueryCondition`**.
Consumer		A system entity invoking producers in a manner conforming to a specification. For example, a portal aggregating content from portlets accessed using the **WSRP** protocol is a type of consumer. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Container		A standard extension mechanism for containers that provides connectivity to enterprise information systems. A connector is specific to an enterprise information system. It consists of a resource adapter and application development tools for enterprise information system connectivity. The resource adapter is plugged in to a container through its support for system-level contracts defined in the Connector architecture. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Content Discovery Service	CDS	**Net-Centric Enterprise Services** (NCES) service that provided a Federated Search capability.
Core Enterprise Services	CES	Ubiquitous, common solution **services** that provide capabilities essential to the operation of the enterprise. Generic information services that apply to any **COI**, provide the basic ability to search the **enterprise** for desired information, and then establish a connection to the desired service. (Source: http://www.defenselink.mil/nii/org/cio/doc/GIG_ES_Core_Enterprise_Services_Strategy_V1-1a.pdf)
Credentials		The information describing the security attributes of a principal. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
CRL Distribution Point	CDP	The location where the **Certificate Authroity** (**CA**) puts the **Certificate Revocation List** (**CRL**) for relying parties to obtain the most current CRL.
Database Management System	DBMS	A system, usually automated and computerized, for managing any collection of compatible, and ideally normalized, data. (Source: http://en.wikipedia.org/wiki/DBMS)

Data-Centric		An approach for the design and implementation of systems, applications, services or software that emphasis the data rather than the operations. It implies that the data is physically separated from the code and consequently can be maintained independently (loose coupling between code and data).
Data-Centric Publish-Subscribe	DCPS	The Data-Centric Publish-Subscribe is a lower level layer of the **DDS** infrastructure that is targeted towards the efficient delivery of the proper information to the proper recipients.
Data Dictionary		A data dictionary is set of metadata that contains definitions and representations of **data elements**. Within the context of a DBMS, a data dictionary is a read-only set of tables and views. The data dictionary may be considered a database in its own right.
Data Distribution Service for Real-Time Systems	DDS	DDS is a recently-adopted OMG standard that is the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems. DDS introduces a virtual Global Data Space where applications can share information by simply reading and writing data-objects addressed by means of an application-defined name (Topic) and a key. DDS features fine and extensive control of QoS parameters, including reliability, bandwidth, delivery deadlines, and resource limits. DDS also supports the construction of local object models on top of the Global Data Space. (Source: OMG Data Distribution Portal, http://portals.omg.org/dds)

Data Element		A data element is an atomic unit of data that has the following: • an identification such as a data element name • a clear data element definition • one or more representation terms • optional enumerated values
Data Element Gallery		The Data Element Gallery is an important component of the Metadata Registry and Clearinghouse. The Data Element Gallery provides its users with access to **data elements** that are commonly used by the Department of Defense such as country codes and U.S. state codes. Users may search the registry, compare data elements, and download an Access database containing the available elements. See the DoD Metadata Registry, http://metadata.dod.mil.
Data Integrity		A measure of the consistency and accuracy of computer data. Integrity can be threatened by hardware problems, power outages, and disk crashes, but most often is threatened by application software or viruses. In a database program, data integrity can be threatened if two users are allowed to update the same item or record at the same time. Record or File Locking, whereby only a single user is allowed access to a given record at any one point in time is one method of ensuring data integrity. (Source: http://www.courts.state.ny.us/ad4/lib/gloss.html#D)
Data Local Reconstruction Layer	DLRL	The Data Local Reconstruction Layer is an optional part of the **DDS** specification that provides a higher level layer allowing for a simpler integration of the DDS into the application layer.
Data Modeling	DM	Modeling is an essential step in understanding the data that will comprise a system. The end products of data modeling can be XML schemas or RDBMS schema definitions. Many COIs create their own data models, such as **C2IEDM** for the **C2** community.

Data Type		A data type is a constraint placed upon the interpretation of data in a type system in computer programming. Common types of data in programming languages include primitive types (such as integers, floating point numbers or characters), tuples, records, algebraic data types, abstract data types, reference types, classes and function types. A data type describes representation, interpretation and structure of values manipulated by algorithms or objects stored in computer memory or other storage device. The type system uses data type information to check correctness of computer programs that access or manipulate the data. (Source: http://en.wikipedia.org/wiki/Data_type)
DDS DataReader		The **DDS** **`DataReader`** acts as a typed (i.e., dedicated to only one application data type) accessor to a subscriber. The **`DataReader`** class allows the application to declare the data it wishes to receive (i.e., make a subscription) and access the data received by the attached **Subscriber**.
DDS DataWriter		A **DDS** **`DataWriter`** acts as a typed (i.e., dedicated to only one application data type) accessor to a publisher. The **`DataWriter`** class allows the application to set the value of the data to be published under a given **Topic**.
DDS DomainParticipant		A **DDS** domain participant represents the local membership of the computer process in a domain. A **domain** is a distributed concept that links all the computer processes able to communicate with each other. It represents a communication plane; only the **publishers** and the **subscribers** attached to the same domain may interact. A computer process can run on the behalf of some user or application.
DDS Global Data Space		Underlying any **data-centric** publish subscribe system is a data model. In **DDS**, this model defines the global data space and specifies how **Publishers** and **Subscribers** refer to portions of this space. (See DDS **Domain**)
DDS Listener		A **DDS** **`Listener`** is used to provide a callback for synchronous access. Listeners provide a generic mechanism for the middleware to notify the application of relevant asynchronous events, such as arrival of data corresponding to a subscription, violation of a **QoS** setting, etc. Each **DCPS** entity supports its own specialized kind of listener. **`Listener`** operations are invoked using a middleware-provided thread.
DDS Publication		A **DDS** publication is defined by the association of a **DataWriter** to a **publisher**. This association expresses the intent of the application to publish the data described by the DataWriter in the context provided by the publisher.
DDS Publisher		A **DDS** publisher is an object responsible for data distribution. It may publish data of different data types. The **DataWriter** is the object the application must use to communicate to a publisher the existence and value of data-objects of a given type. When data-object values have been communicated to the publisher through the appropriate **`DataWriter`**, it is the publisher's responsibility to perform the distribution (the

		publisher will do this according to its own QoS, or the **QoS** attached to the corresponding **`DataWriter`**).
DDS Subscriber		A **DDS** subscriber is an object responsible for receiving published data and making it available (according to the Subscriber's **QoS**) to the receiving application. It may receive and dispatch data of different specified types. To access the received data, the application must use a typed **DataReader** attached to the subscriber.
DDS Subscriber Access API		**DDS** defines two **APIs** that provide subscriber access: **Listeners** and the dual **Condition/WaitSet** infrastructure allow applications to be notified when changes occur in a **DCPS** communication.
DDS Subscription		A **DDS** subscription is defined by the association of a **DataReader** with a subscriber. This association expresses the intent of the application to subscribe the data described by the **`DataReader`** in the context provided by the **subscriber**.
DDS WaitSet		A **DDS** **`WaitSet`** associated with one or several **Condition** objects provides asynchronous data access. **`WaitSets`** and their associated **`Conditions`** provide the means for an application thread to block waiting for the same events that can be received via a **Listener**. Using a **`WaitSet`** the application can handle the event in its own thread instead of the middleware provided thread used for **`Listeners`**.
Defense Acquisition University	DAU	The mission of the DAU is to provide practitioner training, career management, and services to enable the DoD Acquisition, Technology & Logistics (AT&L) community to make smart business decisions and deliver timely and affordable capabilities to the warfighter. (Source: http://www.dau.mil/about-dau/docs/mission_vision.ppt)
Defense Information System Network	DISN	The Defense Information System Network (DISN) has been the Department of Defense's enterprise network for providing data, video and voice services for more than 40 years. (Source: http://www.disa.mil/main/support/dss.html)
Defense Information Systems Agency	DISA	Combat support agency responsible for planning, engineering, acquiring, fielding, and supporting global net-centric solutions to serve the needs of the President, Vice President, the Secretary of Defense, and other DoD Components, under all conditions of peace and war. (Source: http://www.disa.mil/main/about/missman.html)
Defense IT Standards Registry	DISR	The DoD IT Standards Registry (DISR) is an online repository (http://disronline.disa.mil) for a minimal set of primarily commercial IT standards formerly captured in the Joint Technical Architecture (JTA), Version 6.0. These standards are used as the "building codes" for all systems being procured in the Department of Defense. Use of these building codes facilitates interoperability among systems and integration of new systems into the Global Information Grid (GIG). In addition, the DISR provides the capability to build profiles of standards that programs will use to deliver

		net-centric capabilities. (Source: http://akss.dau.mil/dag/GuideBook/IG_c7.2.4.2.asp)
Department of Defense	DoD	A civilian Cabinet organization of the United States government. The Department of Defense controls the U.S. military and is headquartered at The Pentagon. It is headed by the Secretary of Defense. (Source: http://en.wikipedia.org/wiki/United_States_Department_of_Defense)
Deployment Descriptor		An XML file provided with each module and J2EE application that describes how they should be deployed. The deployment descriptor directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Deprecate		Deprecation is the gradual phasing-out of features such as guidance, software or programming language features. Guidance, features or methods marked as deprecated are considered obsolete, and further use is discouraged. The guidance features or methods are still valid although error messages as warnings may occur when they are referenced. These serve to alert the user to the fact that the feature may be removed in future releases. Features get marked as deprecated, rather than simply removed, in order to provide backward compatibility end users.
Digest		A cryptographic checksum of an octet stream.
Digital Signature		A value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the signature to verify that the data has not been altered and/or has originated from the signer of the message, providing message integrity and authentication. The signature can be computed and verified with symmetric key algorithms, where the same key is used for signing and verifying, or with asymmetric key algorithms, where different keys are used for signing and verifying (a private and public key pair are used).
Digital Signature Algorithm	DSA	The Digital Signature Algorithm (DSA) is a United States Federal Government standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in **Federal Information Processing Standard** (**FIPS**) 186, adopted in 1993. A minor revision was issued in 1996 as FIPS 186-1, and the standard was expanded further in 2000 as FIPS 186-2. (Source: http://en.wikipedia.org/wiki/Digital_Signature_Algorithm)

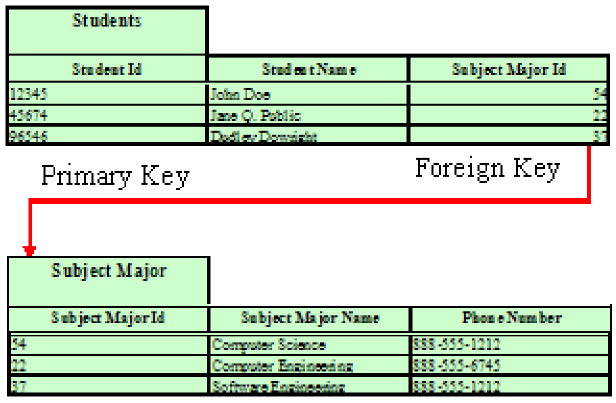
Directory Service		A directory service organizes computerized content and runs on a directory server computer. It is not to be confused with the directory itself, which is the database that holds the information about objects that are to be managed by the directory service. The directory service is the interface to the directory and provides access to the data that is contained in that directory. It acts as a central authority that can securely authenticate resources and manage identities and relationships between them. (Source: http://en.wikipedia.org/wiki/Directory_service)
Doctrine, Organization, Training, Materiel, Leadership, Personnel, Facilities	DOTMLPF	
Document Object Model	DOM	An API for accessing and manipulating XML documents as tree structures. DOM provides platform-neutral, language-neutral interfaces that enable programs and scripts to dynamically access and modify content and structure in XML documents. (Source: *J2EE 1.4 Glossary*, http://java.sun.com/j2ee/1.4/docs/glossary.html)
Document Type Definition	DTD	An optional part of the XML document prolog, as specified by the XML standard. The DTD specifies constraints on the tags and tag sequences that can be in the document. The DTD has a number of shortcomings, however, and this has led to various schema proposals. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
DoD Architecture Framework	DoDAF	Defines a common approach for DoD architecture description, development, presentation, and integration for both warfighting operations and business processes [DoDAF v1.0 supersedes **C4ISR** Architecture Framework v2.0, 18 December 1997]. (Source: Office of the Secretary of Defense memo of 9 Feb 2004, *The Department of Defense Architecture Framework (DoDAF)*)
DoD Discovery Metadata Specification	DDMS	The DoD Discovery Metadata Specification (DDMS) defines discovery metadata elements for resources posted to community and organizational shared spaces. (Source: http://metadata.dod.mil/mdr/irs/DDMS/)

DoD Metadata Registry		As part of the overall **DoD Net-Centric Data Strategy**, the DoD CIO established the DoD Metadata Registry (http://metadata.dod.mil) and a related metadata registration process for the collection, storage and dissemination of structural metadata information resources (schemas, data elements, attributes, document type definitions, style-sheets, data structures, etc.). This Web-based repository is designed to also act as a clearinghouse through which industry and government coordination on metadata technology and related metadata issues can be advanced. As OASD's Executive Agent, **DISA** maintains and operates the ***DoD Metadata Registry and Clearinghouse*** under the direction and oversight of **OASD(NII)**. (Source: DoD Metadata Registry v6.0 Web site, https://metadata.dod.mil/mdr/about.htm)
DoD Net-Centric Data Strategy		This Strategy lays the foundation for realizing the benefits of net-centricity by identifying data goals and approaches for achieving those goals. To realize the vision for net-centric data, two primary objectives must be emphasized: (1) increasing the data that is available to communities or the Enterprise and (2) ensuring that data is usable by both anticipated and unanticipated users and applications. (Source: *Department of Defense Net-Centric Data Strategy*, DoD CIO, 9 May 2003, http://www.defenselink.mil/cio-nii/docs/Net-Centric-Data-Strategy-2003-05-092.pdf)
DoD PKI High Assurance		Applications that handle high value unclassified information (mission critical) in minimally protected environments require High Assurance certificates. Applications that are applicable for High Assurance certificates include the following: • All applications appropriate for DoD PKI Medium Assurance certificates • Digital signature services for unclassified Mission Assurance Category I (MAC I) or national security information in an unencrypted network • Protection (authentication and confidentiality) for information crossing classification boundaries when such a crossing is already permitted under a system security policy (e.g., sending unclassified information through a High Assurance Guard from **SIPRNet** to **NIPRNet**) (Source: adapted from ***X.509 Certificate Policy for the United States Department of Defense***, Version 9.0, 9 February 2005; http://iase.disa.mil/pki/dod-cp-v90-final-9-feb-05-signed.pdf; DoD PKI Certificate required)
Domain		A group of related items within a certain area of interest. In **DDS**, a domain is the basic construct used to bind individual **publications** and **subscriptions** together for communication. A distributed application can elect to use single or multiple domains for its **data-centric** communications. Domains isolate communication, promote scalability and segregate different classifications of data. (See **Global Data Space**)

Domain Analysis		The process of identifying the types of information that the data model uses. A good data model captures descriptive information about each of the types.
Domain Name System	DNS	The Domain Name System stores information about hostnames and domain names in a type of distributed database on networks, such as the Internet. Of the many types of information that can be stored, most importantly it provides a physical location (IP address) for each domain name, and lists the mail exchange servers accepting email for each domain. The DNS provides a vital service on the Internet as it allows the transmission of technical information in a user-friendly way. While computers and network hardware work with IP addresses to perform tasks such as addressing and routing, humans generally find it easier to work with hostnames and domain names (such as ***www.example.com***) in URLs and email addresses. The DNS therefore mediates between the needs and preferences of humans and of software.
Dual Stacking		Incorporating both IPv4 and IPv6 support in routers and computers.
Dynamic Host Configuration Protocol	DHCP	A protocol for assigning dynamic **Internet Protocol** (IP) addresses to devices on a network; DHCP a device can have a different IP address every time it connects to the network. (Source: http://www.webopedia.com/TERM/D/DHCP.html)
Encryption		Encryption is the process of obscuring information to make it unreadable without special knowledge. While encryption has been used to protect communications for centuries, only organizations and individuals with an extraordinary need for secrecy have made use of it. In the mid-1970s, strong encryption emerged from the sole preserve of secretive government agencies into the public domain, and is now employed in protecting widely-used systems, such as Internet e-commerce, mobile telephone networks and bank automatic teller machines. (Source: http://en.wikipedia.org/wiki/Encryption)
Endpoint		The URL or location of the Web service on the internet.
Enterprise		An organization considered as an entity or system that includes interdependent resources (e.g., people, organizations, and technology) that must coordinate functions and share information in support of a common mission or a set of related missions. In the computer industry, the term is often used to describe any large organization that utilizes computers. An intranet, for example, is a good example of an enterprise computing system. (Source: http://www.webopedia.com/TERM/e/enterprise.html)
Enterprise Java Bean	EJB	A server-side component architecture for the development and deployment of object-oriented, distributed, enterprise-level applications. Applications written using the Enterprise JavaBeans architecture are scalable, transactional,

		and secure. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Enterprise Service		A service that provides capabilities to the enterprise. See also **Core Enterprise Service** and **Community of Interest Service**.
Environment Variable		Environment variables are a set of dynamic values that can affect the way running processes will behave. (Source: http://en.wikipedia.org/wiki/Environment_variable)
eXtensible Access Control Markup Language	XACML	XACML is used to represent and evaluate access control policies. XACML is designed to standardize the use of declarative policy to control access to resources. Used with **SAML**.
eXtensible Markup Language	XML	A markup language defines tags (markup) to identify the content, data, and text in XML documents. It differs from **HTML**, the markup language most often used to present information on the Internet. HTML has fixed tags that deal mainly with style or presentation. An XML document must undergo a transformation into a language with style tags under the control of a style sheet before it can be presented by a browser or other presentation mechanism. Two types of style sheets used with XML are CSS and XSL. Typically, XML is transformed into HTML for presentation. Although tags can be defined as needed in the generation of an XML document, you can use a document type definition (DTD) to define the elements allowed in a particular type of document. A document can be compared by using the rules in the DTD to determine its validity and to locate particular elements in the document. A Web services application's J2EE deployment descriptors are expressed in XML with schemas defining allowed elements. Programs for processing XML documents use SAX or DOM APIs. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
eXtensible Stylesheet Language	XSL	Extensible Stylesheet Language (XSL) is a family of recommendations for defining XML document transformation and presentation. It consists of three parts: • XSL Transformations (XSLT): a language for transforming XML • XML Path Language (XPath): an expression language used by XSLT to access or refer to parts of an XML document • XSL Formatting Objects (XSL-FO): an XML vocabulary for specifying formatting semantics (Source: http://www.w3.org/Style/XSL/)

Facade		Provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use. This can simplify a number of complicated object interactions into a single interface.
Facade Design Pattern		An object that provides a simplified interface to a larger body of code, such as a class library. (Source: http://en.wikipedia.org/wiki/Facade_pattern)
Federal Information Processing Standard	FIPS	Under the Information Technology Management Reform Act (Public Law 104-106), the Secretary of Commerce approves standards and guidelines that are developed by the National Institute of Standards and Technology (NIST) for Federal computer systems. These standards and guidelines are issued by NIST as Federal Information Processing Standards (FIPS) for use government-wide. NIST develops FIPS when there are compelling Federal government requirements such as for security and interoperability and there are no acceptable industry standards or solutions. (Source: http://www.itl.nist.gov/fipspubs/geninfo.htm)
Federated Search		Implementation of a computer program that allows users to access multiple data sources with a single query string located within a single interface. (Source: http://en.wikipedia.org/wiki/Federated_search)
File Transfer Protocol	FTP	FTP transfers files to and from a remote network. The protocol includes the ftp command (local machine) and the in.ftpd daemon (remote machine). FTP enables a user to specify the name of the remote host and file transfer command options on the local host's command line. The in.ftpd daemon on the remote host then handles the requests from the local host. Unlike RCP, FTP works even when the remote computer does not run a UNIX-based operating system. A user must log in to the remote computer to make an FTB connection unless it has been set up to allow anonymous FTP. (Source: http://www.sun.com/products-n-solutions/hardware/docs/html/817-6210-10/glossary.html)
Firewall		A piece of hardware and/or software which functions in a networked environment to prevent some communications forbidden by the security policy, analogous to the function of firewalls in building construction.
Font Size		The font size refers to the size of the font from baseline to baseline, when set solid (in CSS terms, this is when the ***font-size*** and ***line-height*** properties have the same value). (Source: http://www.w3.org/TR/REC-CSS2/fonts.html)
Foreign Key	FK	An attribute in a relation of a database that serves as the primary key of another relation in the same database.

		 Students Student Id | Student Name | Subject Major Id 12345 | John Doe | 54 45674 | Jane Q. Public | 22 96546 | Dudley Dowright | 37 Subject Major Subject Major Id | Subject Major Name | Phone Number 54 | Computer Science | 888-555-1212 22 | Computer Engineering | 888-555-6745 37 | Software Engineering | 888-555-1212 I1156
GIG Enterprise Service		A service that provides capabilities for use in the DoD enterprise. GIG Enterprise Services are the combination of Core Enterprise Services and Community of Interest Services. Also referred to as Global Enterprise Services.
Global Command and Control System	GCCS	GCCS-J is the DOD joint C2 system of record for achieving full spectrum dominance. It enhances information superiority and supports the operational concepts of full-dimensional protection and precision engagement. GCCS-J is the principal foundation for dominant battlespace awareness, providing an integrated, near real-time picture of the battlespace necessary to conduct joint and multinational operations. It fuses select C2 capabilities into a comprehensive, interoperable system by exchanging imagery, intelligence, status of forces, and planning information. GCCS-J offers vital connectivity to the systems the joint warfighter uses to plan, execute, and manage military operations. GCCS-J is a Command, Control, Communications, Computer, and Intelligence (C4I) system, consisting of hardware, software, procedures, standards, and interfaces that provide a robust, seamless C2 capability. The system uses the Defense Information Systems Network (DISN) and must work over tactical communication systems to ensure connectivity with deployed forces in the tactical environment. (Source: http://www.disa.mil/gccs-j/)
Global Information Grid	GIG	Globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, policy makers, and support personnel. The GIG includes all owned and leased communications and computing systems and services, software (including applications), data, security services, and other associated services necessary to achieve Information Superiority. It also includes National Security Systems (NSS) as defined in section 5142 of the Clinger-Cohen Act of 1996. The GIG supports all DoD, National Security, and related Intelligence Community (IC) missions and functions (strategic, operational, tactical, and business) in war and in peace. The GIG provides capabilities from all operating locations (bases, posts, camps, stations, facilities, mobile platforms,

		and deployed sites). The GIG provides interfaces to coalition, allied, and non-DoD users and systems.
Global Positioning System		A satellite constellation that provides highly accurate position, velocity, and time navigation information to users. (Source: JP 1-02, http://www.dtic.mil/doctrine/jel/doddict/data/g/02300.html)
Graphical User Interface	GUI	A program that lets the user interact with a computer system in a highly visual manner, with a minimum of typing. Graphical user interfaces usually require a high-resolution display and a pointing device, such as a computer mouse. (Source: http://www.oreilly.com/catalog/debian/chapter/book/glossary.html)
Hard Code		To hard code or hard coding (also, hard-code/hard-coding, hardcode/hardcoding) refers to the software development practice of embedding output or configuration data directly into the source code of a program or other executable object, or fixed formatting of the data, instead of obtaining that data from external sources or generating data or formatting in the program itself with the given input. Considered an ***anti-pattern*** or ***Bad Thing***, hard coding requires the program's source code to be changed any time the input data or desired format changes, when it might be more convenient to the end user to change the detail by some means outside the program. (Source: http://en.wikipedia.org/wiki/Hard_code; 12 June 2007)
High Assurance Internet Protocol Encryption	HAIPE	DoD version of Internet Protocol (IP) security (IPsec) protocol. (Source: http://en.wikipedia.org/wiki/HAIPE)
High Availability		Data tier availability can be affected by hardware failure, power outages, data errors, user errors, programmer errors, OS errors, and RDBMS errors. Various hardware and software methods help mitigate availability issues. The more reliable a system needs to be, the more it costs. Consequently, defining availability to meet requirements is essential to controlling costs.
Horizontal Fusion	HF	Horizontal Fusion (HF) is a direct response to Secretary of Defense Donald H. Rumsfeld's vision of Force Transformation. It demonstrates the ability to use lightweight automation to replace system mass with superior access to information based on a coherent architecture for an arbitrary future. Horizontal Fusion acts as a catalyst by implementing and demonstrating technologies and techniques that significantly advance the process of information-sharing in a an evolving net-centric environment. (Source: http://horizontalfusion.dtic.mil/vision/)
Hypertext Markup Language	HTML	A markup language for hypertext documents on the Internet. HTML supports embedding images, sounds, video streams, form fields, references to other objects with URLs, and basic text formatting. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Hypertext Transfer Protocol	HTTP	The Internet protocol used to retrieve hypertext objects from remote hosts. HTTP messages consist of requests from client

		to server and responses from server to client. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Hypertext Transmission Protocol Over SSL	HTTPS	HTTPS is the secure version of **HTTP**, the communication protocol of the World Wide Web. It was invented by Netscape Communications Corporation to provide authentication and encrypted communication and is used in electronic commerce. Instead of using plain text socket communication, HTTPS encrypts the session data using either a version of the **SSL** (Secure Socket Layer) protocol or the **TLS** (Transport Layer Security) protocol, thus ensuring reasonable protection from eavesdroppers, and man in the middle attacks. The default TCP/IP port of HTTPS is 443. (Source: http://en.wikipedia.org/wiki/HTTPS)
Identity		Identity refers to the nature or attributes of the track: ***Friend***, ***Assumed Friend***, ***Neutral***, ***Unknown***, ***Pending***, ***Suspect***, or ***Hostile***.
Image Map		An image or graphic that has been coded to contain interactive areas. When it is clicked on, it launches another Web page or program. An image map usually has many different hyperlinked areas, known as links. For example, an image map of a country could be coded so that when a user clicks on a city or region, the browser is routed to a document or Web page about that place. (Source: http://www.netlingo.com/right.cfm?term=clickable%20graphic%20or%20imagemap)
Information Assurance	IA	Measures taken to protect and defend our information and information systems to ensure Confidentiality, Integrity, Availability, and Accountability, extended to restoration with protect, detect, monitor, and react capabilities.
Information Technology	IT	Any equipment or interconnected system or subsystem of equipment, that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information. Information technology includes computers, ancillary equipment, software, firmware, and similar procedures, services (including support services), and related resources. Information technology does not include any equipment that is acquired by a federal contractor incidental to a federal contract. (Source: CJCSI 6212.01D, 8 March 2006, Glossary page GL-11)
Initial Capabilities Document	ICD	Documents the need for a materiel approach, or an approach that is a combination of materiel and non-materiel, to satisfy specific capability gap(s). It defines the capability gap(s) in terms of the functional area, the relevant range of military operations, desired effects, time and doctrine, organization, training, materiel, leadership and education, personnel, and facilities (DOTMLPF) and policy implications and constraints. The ICD summarizes the results of the DOTMLPF and policy analysis and the DOTMLPF approaches (materiel and non-materiel) that may deliver the required capability.

		The outcome of an ICD could be one or more joint DCRs or capability development documents. (Source: CJCSI 3170.01E, *Joint Capabilities Integration and Development System*, 11 May 2005, Glossary page GL-8)
Integrated Development Environment	IDE	
Interface		The functional and physical characteristics required to exist at a common boundary or connection between systems or items. (Source: DoD 4120.214-M)
Interface Definition Language	IDL	A language used to define interfaces to remote **CORBA** objects. The interfaces are independent of operating systems and programming languages. (Source: http://java.sun.com/javaee/reference/glossary/index.jsp#120354)
International Telecommunication Union	ITU	United Nations agency for information and communication technologies. (Source: http://www.itu.int/net/about/index.aspx)
Internet		The Internet, or simply the Net, is the publicly available worldwide system of interconnected computer networks that transmit data by packet switching using a standardized Internet Protocol (IP) and many other protocols. It is made up of thousands of smaller commercial, academic, and government networks. It carries various information and services, such as electronic mail, online chat and the interlinked web pages and other documents of the World Wide Web. Because this is by far the largest, most extensive internet (with a lower case i) in the world, it is simply called the Internet (with a capital I). (Source: http://en.wikipedia.org/wiki/Internet)
Internet Engineering Task Force	IETF	The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual. (Source: http://www.ietf.org/overview.html)
Internet Information Services	IIS	A set of Internet-based services for Windows machines. Originally supplied as part of the Option Pack for Windows NT, they were subsequently integrated with Windows 2000 and Windows Server 2003. The current (Windows 2003) version is IIS 6.0 and includes servers for FTP, SMTP, NNTP and HTTP/HTTPS. Earlier versions also included a Gopher server.
Internet Protocol	IP	Data packets routed across network, not switched via dedicated circuits.
Internet Protocol Version 4	IPv4	Version 4 of the Internet Protocol (IP). It was the first version of the Internet Protocol to be widely deployed, and forms the basis for most of the current Internet (as of 2004). It is described in IETF RFC 791, which was first published in September, 1981. IPv4 uses 32-bit addresses, limiting it to 4,294,967,296 unique addresses, many of which are reserved for special purposes such as local networks or **multicast**

		addresses. This reduces the number of addresses that can be allocated as public Internet addresses. As the number of addresses available is consumed, an IPv4 address shortage appears to be inevitable in the long run. This limitation has helped stimulate the push towards IPv6, which is currently in the early stages of deployment, and may eventually replace IPv4. (Source: http://en.wikipedia.org/wiki/IPv4)
Internet Protocol Version 6	IPv6	Version 6 of the Internet Protocol; it was initially called IP Next Generation (IPng) when it was picked as the winner in the IETF's IPng selection process. IPv6 is intended to replace the previous standard, IPv4, which only supports up to about 4 billion (4 x 10^9) addresses. IPv6 supports up to about 3.4 x 10^{38} (340 undecillion) addresses. This is the equivalent of 4.3 x 10^{20} (430 quintillion) addresses per square inch (6.7 x 10^{17} (670 quadrillion) addresses/mm^2)of the Earth's surface. It is expected that IPv4 will be supported until at least 2025, to allow time for bugs and system errors to be corrected. (Source: http://en.wikipedia.org/wiki/Ipv6)
Interoperability		The ability of systems, units, or forces to provide data, information, materiel, and services to and accept the same from other systems, units, or forces, and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together. **IT** and **NSS** interoperability includes both the technical exchange of information and the end-to-end operational effectiveness of that exchanged information as required for mission accomplishment. Interoperability is more than just information exchange. It includes systems, processes, procedures, organizations, and missions over the life cycle and must be balanced with information assurance. (Source: CJCSI 6212.01D, ***Interoperability and Supportability of Information Technology and National Security Systems***, 8 March 2006)
J2EE Server		The runtime portion of a J2EE product. A J2EE server provides EJB or Web containers or both. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Java 2 Platform, Enterprise Edition	J2EE	The J2EE environment is the standard for developing component-based multi-tier enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, Web-based applications. Features include Web services support and development tools. Sun Microsystems has simplified the name of the Java platform for the enterprise; the "2" is dropped from the name, as well as the dot number so the next version of the Java platform for the enterprise is **Java Platform, Enterprise Edition** 5 or Java EE 5.(Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Java Archive	JAR	A platform-independent file format that enables you to bundle multiple files into a single archive file. JAR files are packaged with the ZIP file format, so you can use them for ZIP-like tasks such as lossless data compression, archiving, decompression, and archive unpacking. Typically JAR files

		contain the class files and auxiliary resources associated with applets and applications. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Java Database Connection	JDBC	An API that supports database and data-source access from Java applications.
Java Development Kit	JDK	
Javadoc		Javadoc is a computer software tool from Sun Microsystems for generating API documentation into HTML format from Java source code. Javadoc is the industry standard for documenting Java classes. Most **Integrated Development Environments** (**IDEs**) will automatically generate Javadoc HTML. (Source: http://en.wikipedia.org/wiki/Javadoc)
Java Message Service	JMS	An API for invoking operations on enterprise messaging systems. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Java Naming and Directory Interface	JNDI	An API that provides naming and directory functionality. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Java Platform, Enterprise Edition	Java EE	Java Platform, Enterprise Edition (Java EE) is the industry standard for developing portable, robust, scalable and secure server-side Java applications. Building on the solid foundation of the Java Platform, Standard Edition (Java SE), Java EE provides Web services, component model, management, and communications APIs that make it the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation Web applications. Sun Microsystems has simplified the name of the Java platform for the enterprise. Formerly, the platform was known as Java 2 Platform, Enterprise Edition (**J2EE**), and specific versions had "dot numbers" such as J2EE 1.4. The "2" is dropped from the name, as well as the dot number so the next version of the Java platform for the enterprise is Java Platform, Enterprise Edition 5 or Java EE 5. (Source: http://java.sun.com/javaee/)
JavaScript		The Netscape-developed object scripting language used in millions of web pages and server applications worldwide. Contrary to popular misconception, JavaScript is not "Interpretive Java." Rather, it is a dynamic scripting language that supports prototype-based object construction.
JavaServer Page	JSP	An extensible Web technology that uses static data, JSP elements, and server-side Java objects to generate dynamic content for a client. Typically the static data is HTML or XML elements, and in many cases the client is a Web browser. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Joint Capabilities Integration and Development System	JCIDS	Establishes procedures to support the Chairman of the Joint Chiefs of Staff and the Joint Requirements Oversight Council (JROC) in identifying, assessing and prioritizing joint military capability. (Source: CJCSI 3170.01E, 11 May 2005, *Joint Capabilities Integration and Development System*)

Joint Interoperability Test Command	JITC	Independent operational test and evaluation/assessor of DISA and other DoD Command, Control, Communications, Computers and Intelligence (C4I) acquisitions. (Source: http://jitc.fhu.disa.mil/mission.htm)
Joint Tactical Radio System	JTRS	JTRS is a family of interoperable, affordable software defined radios at moderate risk which provide secure, wireless networking communications capabilities for Joint forces. (Source: JTRS JPEO, http://enterprise.spawar.navy.mil/body.cfm?type=ccategory=27subcat=60)
Joint Worldwide Intelligence Communications System	JWICS	The sensitive, compartmented information portion of the **Defense Information Systems Network**. It incorporates advanced networking technologies that permit point-to-point or multipoint information exchange involving voice, text, graphics, data, and video teleconferencing. (Source: http://www.dtic.mil/doctrine/jel/doddict/data/j/02972.html)
JScript		Microsoft's extended implementation of ECMAScript (ECMA262), an international standard based on Netscape's JavaScript and Microsoft's JScript languages. JScript is implemented as a Windows Script engine. This means that you can plug it in to any application that supports Windows Script, such as Internet Explorer, Active Server Pages, and Windows Script Host. It also means that any application supporting Windows Script can use multiple languages: JScript, VBScript, Perl, and others.
Key Interface Profile	KIP	An operational functionality, systems functionality and technical specifications description of the Key Interface. The profile consists of refined Operational and Systems Views, interface control specifications, Technical View with SV-TV Bridge, and referenced procedures for KIP compliance. The key interface profile is the technical specification that governs access to the **GIG**. (Source: CJCSI 6212.01D, 8 March 2006, Glossary page GL-14)
Key Performance Parameters	KPP	Those attributes or characteristics of a system that are considered critical or essential to the development of an effective military capability and those attributes that make a significant contribution to the key characteristics as defined in the Joint Operations Concepts. KPPs are validated by the Joint Requirements Oversight Council (JROC) for JROC Interest documents, and by the DOD component for Joint Integration or Independent documents. Capability development and capability production document KPPs are included verbatim in the acquisition program baseline. (Source: CJCSI 3170.01E. *Joint Capabilities and Development System*, 11 May 2005, Glossary page GL-12)
Key Recovery Manager	KRM	A service of the DOD PKI where copies of key pairs used for encryption are stored and can be recovered for law enforcement purposes. ***Note:*** *This definition is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*

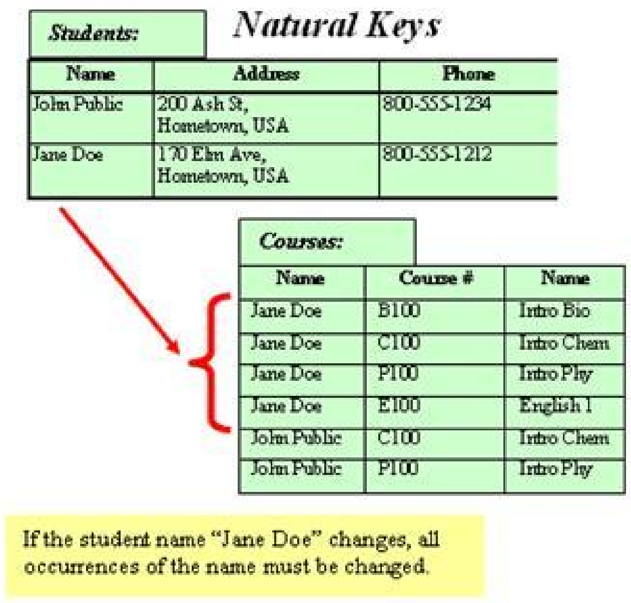
Keystore		A file containing the keys and certificates used for authentication. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Land C2 Information Exchange Data Model	LC2IEDM	
Least-Common-Denominator Data Access Mechanism		When one application is able to obtain data provided by another by removing arbitrary implementation barriers to data exchange.
Legacy System		An existing computer system or application program which continues to be used because the user (typically an organization) does not want to replace or redesign it. (Source: http://en.wikipedia.org/wiki/Legacy_system)
Light Directory Access Protocol	LDAP	A set of protocols for accessing information directories. LDAP is a simpler version of the X.500 standard. Unlike X.500, LD Web Services for Interactive Applications AP supports TCP/IP, which is necessary for Internet access. Because it's a simpler version of X.500, LDAP is sometimes called X.500-lite. LDAP is a protocol for accessing on-line directory services. (Source: http://en.wikipedia.org/wiki/LDAP)
Link-16	TADIL-J	Tactical Data Information Link (TADIL) primarily designed for use by Command and Control (C2) and Air-to-Air assets; uses the Joint Tactical Data Link (TADIL-J) message format. (Source: http://aatc.aztucs.ang.af.mil/aatcinfo.htm)
Linked Style Sheets		Style sheets that are placed in a separate text files and saved in the root with a css file extension. A link to the file is made in the head section of the document. `<head><Break/> <link<Break/> rel="stylesheet"<Break/> href="mystyle.css"<Break/> type="text/css"><Break/></head><Break/>`
Local Area Network	LAN	A group of interconnected computer and support devices. (Source: http://www.sun.com/products-n-solutions/hardware/docs/html/817-6210-10/glossary.html)

Look and Feel		Look and feel refers to design aspects of a graphical user interface in terms of colors, shapes, layout, typefaces, etc. (the "look"); and, the behavior of dynamic elements such as buttons, boxes, and menus (the "feel"). It is used in reference to both software and **Web sites**. (Source: http://en.wikipedia.org/wiki/Look_and_feel)
Loosely Coupled		A computing model where application elements require a simple level of coordination and allow for flexible reconfiguration. Interconnection is often asynchronous and message-based.
Mediation		A set of negotiated agreements for interacting between components that enable those components to work together to perform a task. These agreements are defined through standard interfaces and data interchange specifications. Mediation services provide multiple methods for integrating data sources and services:

Transformation	When a client requests particular format, a tran the data before returnin
Aggregation	A mediator service ma multiple sources, thus be one
Adaptation	When a client cannot c service, an adapter pro transport protocol as w need to communicate p
Orchestration	Co-ordination of events directs and manages th multiple component se application or business
Choreography	When a client request or service requests tha coordinator, a Choreog when to execute other services to interact; WS business process man implements choreogra

Message		A complete unit of data available to be sent or received by services. It is a self-contained unit of information exchange. A message always contains a **SOAP** envelope, and may include additional **MIME** parts as specified in MTOM, and/or transport.
Metadata		Data about the data, that is, the description of the data resources, its characteristics, location, usage, and so on. Metadata is used to identify, describe, and define user data.
Mission		The task, together with the purpose, that clearly indicates the action to be taken and the reason for that action.
Modular Design		Characterized by (1) Functional partitioning into discrete scalable, reusable modules consisting of isolated, self-contained functional elements; (2) Rigorous use of well-

		defined modular interfaces, including object-oriented descriptions of module functionality; (3) Ease of change to achieve technology transparency and, to the extent possible, make use of industry standards for key interfaces.
Module		(1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine. (2) A logically separable part of a program. Note: The terms ***module***, ***component***, and ***unit*** are often used interchangeably or defined to be sub-elements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized. See also **component**. (Source: IEEE Std 610.12-1990)
Multicast		The delivery of information to a group of destinations simultaneously using the most efficient strategy to deliver the messages over each link of the network only once and only create copies when the links to the destinations split. (Source: http://en.wikipedia.org/wiki/Multicast)
Multi-Purpose Internet Mail Extensions	MIME	
MX Record		An MX record or Mail exchanger record is a type of resource record in the **Domain Name System** (DNS) specifying how **Internet** e-mail should be routed. MX records point to the servers that should receive an e-mail, and their priority relative to each other. (Source: http://en.wikipedia.org/wiki/MX_Record)
Namespace		A namespace is an abstract container which contains a logical grouping of unique identifiers (i.e., names). An identifier defined in a namespace is associated with that namespace. It is possible to define the same identifier independently in multiple namespaces. That is, the meaning associated with an identifier defined in one namespace may or may not have the same meaning as the same identifier defined in another namespace. Languages that support namespaces specify the rules that determine to which namespace an identifier (i.e., not its definition) belongs. (Adapted from: http://en.wikipedia.org/wiki/Namespace_%28computer_science%29; accessed 2/6/2008) XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references. (Source http://www.w3.org/TR/REC-xml-names/)
National Security Agency	NSA	America's cryptologic organization; it coordinates, directs, and performs highly specialized activities to protect U.S. government information systems and produce foreign signals intelligence information. (Source: http://www.nsa.gov/about/index.cfm)

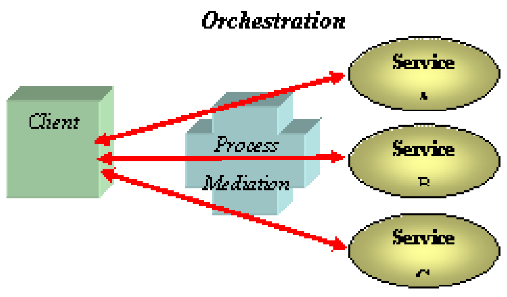
National Security Systems	NSS	Telecommunications and information systems, operated by the Department of Defense, the functions, operation, or use of which involves: (1) intelligence activities; (2) cryptologic activities related to national security; (3) the command and control of military forces; (4) equipment that is an integral part of a weapon or weapons systems; or (5) is critical to the direct fulfillment of military or intelligence missions. Subsection (5) in the preceding sentence does not include procurement of automatic data processing equipment or services to be used for routine administrative and business applications (including payroll, finance, logistics, and personnel management applications). (Source: CJCSI 3170.01F, 1 May 2007, page GL-16)
Natural Key		A Natural Key is a primary keys that is made up completely or in part from naturally occurring data in the tables.  Natural Keys Students: Name | Address | Phone John Public | 200 Ash St, Hometown, USA | 800-555-1234 Jane Doe | 170 Elm Ave, Hometown, USA | 800-555-1212 Courses: Name | Course # | Name Jane Doe | B100 | Intro Bio Jane Doe | C100 | Intro Chem Jane Doe | P100 | Intro Phy Jane Doe | E100 | English 1 John Public | C100 | Intro Chem John Public | P100 | Intro Phy If the student name "Jane Doe" changes, all occurrences of the name must be changed. I1163 See **Surrogate Key** and **Primary Key**.
Net-Centric		Information-based operations that use service-oriented information processing, networks, and data from the following perspectives: user functionality (capability to adaptively perform assigned operational roles with increasing use of system-provided intelligence/cognitive processes), interoperability (shared information and loosely coupled services), and enterprise management (net operations). (Source: DoD Instruction 4630.8, Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS), June 30, 2004 [R1168])
Net-Centric Enterprise Services	NCES	The NCES program provides enterprise-level Information Technology (IT) services and infrastructure components, also called Core Enterprise Services, for the Department of Defense (DoD) Global Information Grid (GIG).

Net-Centric Enterprise Solutions for Interoperability	NESI	A cross service effort between the U.S. Navy Program Executive Office for Command, Control, Communications, Computers and Intelligence (PEO C4I), the U.S. Air Force Electronic Systems Center (ESC) and the Defense Information Systems Agency (DISA). NESI provides a reference architecture, implementation guidance, and a set of reusable software components. These facilitate the design, development, maintenance, evolution, and use of information systems for the Net-Centric Operations and Warfare (NCOW) environment.
Net-Centricity		Net-centricity is an information superiority-enabled concept of operations that generates increased combat power by networking sensors, decision-makers, and shooters to achieve shared awareness, increased speed of command, higher tempo of operations, greater lethality, increased survivability, and a degree of self-synchronization. In essence, net-centricity translates information superiority into combat power by effectively linking knowledgeable entities in the battlespace. (Source: ASD(NII) Net-Centric Checklist v2.1.3, 12 May 2004)
Net-Centric Operations and Warfare Reference Model	NCOW RM	The NCOW RM describes the activities required to establish, use, operate, and manage the net-centric enterprise information environment to include: the generic userinterface, the intelligent-assistant capabilities, the net-centric service capabilities (core services, **Community of Interest (COI) services**, and environment control services), and the enterprise management components. It also describes a selected set of key standards that will be needed as the NCOW capabilities of the **Global Information Grid** (GIG) are realized. The NCOW RM represents the objective end-state for the GIG. This objective end-state is a service-oriented, inter-networked, information infrastructure in which users request and receive services that enable operational capabilities across the range of military operations; **DoD** business operations; and Department-wide enterprise management operations. The NCOW RM is a key compliance mechanism for evaluating DoD information technology capabilities and the **Net-Ready Key Performance Parameter**. (Source: CJCSI 6212.01D, 8 March 2006, Glossary pages GL-17 and GL-18)
Net-Ready Key Performance Parameter	NR-KPP	The NR-KPP assesses information needs, information timeliness, information assurance, and net-ready attributes required for both the technical exchange of information and the end-to-end operational effectiveness of that exchange. The NR-KPP consists of verifiable performance measures and associated metrics required to evaluate the timely, accurate, and complete exchange and use of information to satisfy information needs for a given capability. The NR-KPP is comprised of the following elements: • Compliance with the **NCOW RM**. • Compliance with applicable **GIG KIPs**.

		• Verification of compliance with DoD information assurance requirements. • Supporting integrated architecture products required to assess information exchange and use for a given capability. (Source: DoD Instruction 4630.8, *Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*, 30 June 2004, [R1168] Enclosure 2 Section E2.1.51)
Network Operations	NetOps	An organizational, procedural, and technological construct for ensuring information and decision superiority at the strategic, operational, and tactical levels of warfare as well as within DoD business operations. NetOps is an operational approach, which addresses the interdependency and integration of IA/CND, S&NM, and CS capabilities. NetOps consists of the organizations, tactics, techniques, procedures, functionalities, and technologies required to plan, administer, and monitor use of the GIG infrastructure and the end-to-end information flows of the GIG; and to respond to threats, outages, and other operational impact. NetOps ensures mission requirements are properly considered in GIG operational decision-making. NetOps enables the GIG to provide its users with information they need, when and where they need it, with appropriate protection. NetOps is essential for successful execution of net-centric warfare and other net-centric operations in support of national security objectives.
Network Time Protocol	NTP	Protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. NTP uses **User Datagram Protocol** (UDP) port 123 as its transport layer. It is designed particularly to resist the effects of variable latency. (Source: http://en.wikipedia.org/wiki/Network_Time_Protocol)

Node		In general network usage, a node is a processing location such as a computer or some other device. Every node has a unique network address, sometimes called a Data Link Control (DLC) address or Media Access Control (MAC) address. (Source: http://www.webopedia.com/TERM/n/node.html) A NESI Node is a collection of integrated components (i.e., systems, applications, services and other Nodes) that are bound together spatially and/or temporally to meet the needs of a particular mission. It is conceptual in nature and can not be defined in terms of a concrete set of components or size. The membership of a component within a particular Node is not exclusive and a Component can be part of multiple Nodes.
Nonce		A unique random string.
Normalization		Normalization avoids duplication of data, insert anomalies, delete anomalies, and update anomalies. A relation is in first normal form (1NF) if and only if all underlying simple domains contain atomic values only. A relation is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key. A relation is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Data models should follow the three forms unless there is overriding justification not to. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
North Atlantic Treaty Organization	NATO	NATO is an international organization for defense collaboration established in 1949, in support of the North Atlantic Treaty signed in Washington, D.C., on April 4, 1949. Its other official name is the French equivalent, l'Organisation du Trait de l'Atlantique du Nord (OTAN).
Object Management Group	OMG	OMGTM is an international, open membership, not-for-profit computer industry consortium. OMG Task Forces develop enterprise integration standards for a wide range of technologies, and an even wider range of industries. OMG's modeling standards enable powerful visual design, execution and maintenance of software and other processes. OMG's middleware standards and profiles are based on the **Common Object Request Broker Architecture** (CORBA) and support a wide variety of industries. (Source: http://www.omg.org/)
Object-Oriented Analysis	OOA	OOA (Object Oriented Analysis) constitutes the development of software engineering requirements and specifications for a system. These are expressed as an object model (object oriented design) which is composed of a population of interacting objects.
Object Request Broker	ORB	A library that enables **CORBA** objects to locate and communicate with one another. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)

Online Certificate Status Protocol	OCSP	Online Certificate Status Protocol is a method for determining the revocation status of an X.509 digital certificate using means other than **CRLs**. It is described in RFC 2560 and is on the Internet standards track. OCSP messages are encoded in ASN.1 and usually communicated over **HTTP**. OCSP's request/response nature leads to OCSP servers being termed as OCSP responders.
Online Status Check	OSC	OSC is service that may be provided by the **Certificate Authority** (CA). A relying party sends a request to the OSC service with a certificate, the OSC service responds with a digitally signed response that includes the date and time, certificate identification, and the status of the certificate about whose validity the relying party inquired. The possible responses include "unknown" which may be the response to a query regarding an expired certificate. ***Note:*** *This definition is derived from the DoD Class 3 PKI Public Key-Enabled Application Requirements Document, Version 1.0, 13 July 2000.*
Online Status Check Responder	OSCR	OSCR is the server that responds to a relying party's OSC request.
Open Database Connectivity	ODBC	In computing, Open Database Connectivity (ODBC) provides a standard software API method for using database management systems (DBMS). The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems. (Source: http://en.wikipedia.org/wiki/Odbc; 30 March 2007)
Open Standard		Open standards are publicly available specifications for achieving a specific task. By allowing anyone to obtain and implement the standard, they can increase compatibility between various hardware and software components, since anyone with the necessary technical know-how and resources can build products that work together with those of the other vendors that base their designs on the standard (although patent holders may impose "reasonable and non-discriminatory" royalty fees and other licensing terms on implementers of the standard). Source: http://en.wikipedia.org/wiki/Open_standard) ***Note:*** *NESI restricts the use of the term "standard" to technologies approved by formalized committees that are open to participation by all interested parties and operate on a consensus basis.*
Operational View	OV	The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. DoD missions include both warfighting missions and business processes. The OV contains graphical and textual products that comprise an identification of the operational nodes and elements, assigned tasks and activities, and information flows required between nodes. It defines the types of information exchanged, the frequency

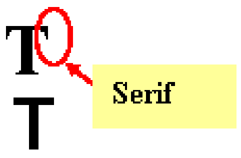
		of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges. (Source: *DoDAF* v1.5 Volume I: Definitions and Guidelines, 23 April 2007)
Orchestration		Co-ordination of events in a process; orchestration directs and manages the on-demand assembly of multiple component services to create a composite application or business process. (Source: http://looselycoupled.com/glossary/orchestration)  Orchestration Client Process Mediation Service A Service B Service C I1164 **Note:** *See* ***Mediation***.
Organization for the Advancement of Structured Information Standards	OASIS	A not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. (Source: http://www.oasis-open.org/who/)
Personal Web Server	PWS	A **Web server** program for personal computer users who want to share **Web pages** and other files from their hard drive. PWS is a scaled-down version of Microsoft's more robust Web server, Internet Information Server (**IIS**). PWS can be used with a full-time **Internet** connection to serve Web pages for a **Web site** with limited traffic. It can also be used for testing a Web site offline or from a "staging" site before putting it on a main Web site that is exposed to more traffic.
Physical Model		Translates the conceptual model to a particular RDBMS implementation.
Portability		The ease with which a system or component can be transferred from hardware or software environment to another. (Source: IEEE Std 610.12-1990) The level of software portability of any specific product depends on two factors: the design of the product itself, and the characteristics of the source and target execution environments. Software products are rarely if ever 100% portable. Generally, the level of portability depends on the target platform. Software that is highly portable to one class of platform might be not portable to other classes.
Portable Object Adapter	POA	A CORBA standard for building server-side applications that are portable across heterogeneous ORBs. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)

Portal		A Web portal is a **Web site** that provides a starting point, gateway, or portal to other resources on the **Internet** or an intranet. Intranet portals are also known as "enterprise information portals" (EIP). Examples of existing portals are Yahoo, Excite, Lycos, Altavista, Infoseek, and Hotbot. (Source: http://en.wikipedia.org/wiki/web_portal)
Portlet		A reusable Web component that displays relevant information to portal users. Examples for portlets include email, weather, discussion forums, and news. The purpose of the **Web Services for Remote Portlets** (WSRP) interface is to provide a **Web services** standard that allows for the "plug-n-play" of **portals**, other intermediary **Web applications** that aggregate content, and applications from disparate sources. The portlet specification enables interoperability between portlets and portals. This specification defines a set of **APIs** for portal computing that addresses the areas of aggregation, personalization, presentation, and security. (Source: http://en.wikipedia.org/wiki/Portlets)
Primary Key	PK	An object that uniquely identifies a row within a table.
Private Key		The private key is one of a pair of keys that are generated as part of asymmetric key cryptography. The private key is kept secret and the public key is public and can be shared openly with others.
Producer		A **Web service** conforming to the **WSRP** specification. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Proxy		A **server** that sits between a client application, such as a **Web browser**, and a real server. It intercepts all requests to the real server to see if it can fulfill the requests itself. If not, it forwards the request to the real server.Proxy servers have two main purposes: improve performance and filter requests. (Source: http://www.webopedia.com/TERM/p/proxy_server.html)
Proxy Pattern		Provides a surrogate or placeholder for another object to control access to it.
Public Key	PK	See **Public Key Cryptography**.

Public Key Certificate		Used in client-certificate authentication to enable the server, and optionally the client, to authenticate each other. The public key certificate is the digital equivalent of a passport. It is issued by a trusted organization, called a certificate authority, and provides identification for the bearer. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Public Key Cryptography		Public key cryptography, also known as asymmetric cryptography, is a form of cryptography in which a user has a pair of cryptographic keys - a public key and a private key. The private key is kept secret, while the public key may be widely distributed. The keys are related mathematically, but the private key cannot be practically derived from the public key. A message encrypted with the public key can be decrypted only with the corresponding private key. (Source: http://en.wikipedia.org/wiki/Public_key)
Public Key Enabling	PK-Enabling	The incorporation of the use of certificates for security services such as authentication, confidentiality, data integrity, and nonrepudiation. PK-Enabling involves replacing existing or creating new user authentication systems using certificates instead of other technologies, such as userid and password or **Internet Protocol** filtering; implementing public key technology to digitally sign, in a legally enforceable manner, transactions and documents; or using public key technology, generally in conjunction with standard symmetric encryption technology, to encrypt information at rest and/or in transit. (Source: DoD Instruction 8520.2, *Public Key Infrastructure (PKI) and Public Key (PK) Enabling*, 1 April 2004 [R1206])
Public Key Infrastructure	PKI	Framework established to issue, maintain, and revoke public key certificates accommodating a variety of security technologies, including the use of software. (Source: CNSS Instruction No. 4009, Revised May 2003, *National Information Assurance (IA) Glossary*)
Quality of Service	QoS	Data timeliness, accuracy, completeness, integrity, and ease of use. Refers to the probability of the network meeting a given traffic contract. In many cases is used informally to refer to the probability of a packet passing between two points in the network. (Source: http://en.wikipedia.org/wiki/Quality_of_service) -OR- A defined level of performance that adapts to the environment in which it is operating. QoS may be requested by the user of the information. The level of QoS provided is based on the request, the available capabilities of the provider, and the priority of the user.

Real-Time		An operation within a larger dynamic system is called a real-time operation if the combined reaction- and operation-time of a task is shorter than the maximum delay that is allowed, in view of circumstances outside the operation. The task must also occur before the system to be controlled becomes unstable. A real-time operation is not necessarily fast, as slow systems can allow slow real-time operations. This applies for all types of dynamically changing systems. The polar opposite of a real-time operation is a batch job with interactive timesharing falling somewhere in-between the two extremes. (Source: http://en.wikipedia.org/wiki/Real_time)
Reference Data Set		The Reference Data Set Gallery [of the **DoD Metadata Registry and Clearinghouse**] provides collections of related data that represent a defined entity within a community of interest. Examples of reference data sets include country codes, U.S. state codes, and marital status codes. (Soure: http://www.disa.mil/nces/development/developer_doc_overview.html)
Referential Integrity		A feature provided by RDBMSs that prevents users or applications from entering inconsistent data. Most RDBMSs have various referential integrity rules that you can apply when you create a relationship between two tables.
Registered Namespace		A namespace that has been registered and approved with a **namespace** registration services. For the DoD, use the **DoD Metadata Registry**.
Registration Web Service	RWS	**Horizontal Fusion** (HF) **service** used by data producers to register content sources.
Relational Database	RDB	A collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.
Relational Database Management System	RDBMS	A database management system (DBMS) that is based on the relational model or that presents the data to the user as relations. A collection of tables, each table consisting of a set of rows and columns, can satisfy this property. RDBMSs also provide relational operators to manipulate the data in tabular form. (Source: http://en.wikipedia.org/wiki/RDBMS)
Relative Font Size		Fonts that display according to the size of the surrounding text. Some designers call them scalable fonts. Instead of displaying a fixed pixel size, a relative font size displays as a percentage of the surrounding elements. (Source: http://www.netmechanic.com/news/vol5/design_no13.htm)
Role-Based Access Control	RBAC	An approach to restricting system access to authorized users. It is a newer and alternative approach to discretionary access control and mandatory access control. It assigns permissions to specific operations with meaning in the organization, rather than to low-level data objects. (Source: http://en.wikipedia.org/wiki/RBAC)

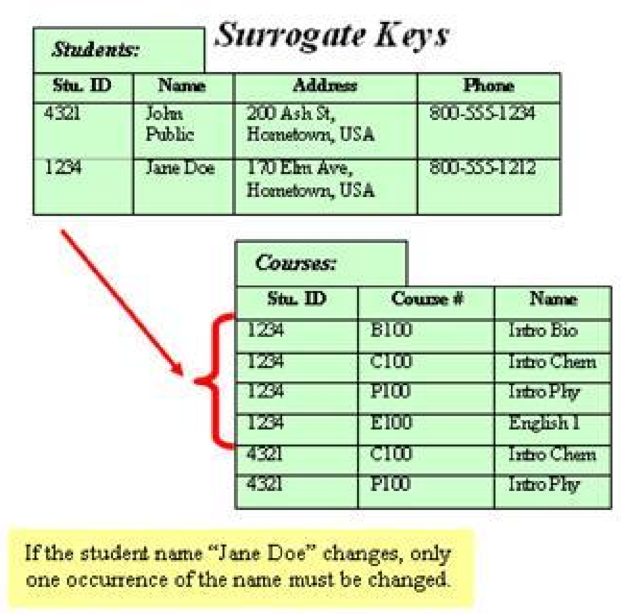
Router		A device that forwards data packets along networks. A router is connected to at least two networks, commonly two **local area networks** (LANs) or wide area networks (WANs) or a LAN and its Internet Service Provider's network. Routers are located at gateways, the places where two or more networks connect. (Source: http://www.webopedia.com/TERM/r/router.html)
SCA Operating Environment	OE	**SCA** Operating Environment: The SCA OE describes the requirements of the operating system, middleware, and the CF interfaces and operations.
Schema		A diagrammatic representation, an outline, or a model. In relation to data management, a schema can represent any generic model or structure that deals with the organization, format, structure, or relationship of data. Some examples of schemas are (1) a database table and relational structure, (2) a **document type definition** (DTD), (3) a data structure used to pass information between systems, and (4) an **XML schema document** (XSD) that represents a data structure and related information encoded as XML. Schemas typically do not contain information specific to a particular instance of data (Source: DoD 8320.02-G, 12 April 2006, *Guidance for Implementing Net-Centric Data Sharing*)
Search Web Service	SWS	**Horizontal Fusion** (HF) **service** used to search for content from registered sources.
Secret Internet Protocol Router Network	SIPRNet	DoD's largest interoperable command and control data network, supporting the **Global Command and Control System** (GCCS), the Defense Message System (DMS), collaborative planning and numerous other classified warfighter applications. Direct connection data rates range from 56 kbps to 155 Mbps for the **Unclassified but Sensitive Internet Protocol Router Network** (NIPRNet), and up to 45 Mbps for the SIPRNet. Remote dial-up services are also available, ranging from 19.2 kbps on SIPRNet to 56 kbps on NIPRNet. (Source: http://www.disa.mil/main/prodsol/data.html)
Secure Hash Algorithm	SHA	The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions. In cryptography, a cryptographic hash function is a hash function with certain additional security properties to make it suitable for use as a primitive in various information security applications, such as authentication and message integrity. A hash function takes a long string (or message) of any length as input and produces a fixed length string as output, sometimes termed a message **digest** or a digital fingerprint. (Source: http://en.wikipedia.org/wiki/SHA#SHA-0_and_SHA-1)
Secure Sockets Layer	SSL	A protocol for transmitting private documents via the Internet. SSL uses a cryptographic system employing two keys to encrypt data: a public key known to everyone and a private or secret key known only to the recipient of the message. (Source:http://www.webopedia.com/TERM/S/SSL.html)

Security Assertion Markup Language	SAML	An XML standard for exchanging authentication and authorization data between security domains; that is, between an identity provider and a service provider. SAML is a product of the **OASIS** Security Services Technical Committee. (Source:http://en.wikipedia.org/wiki/SAML)
Security Technical Implementation Guide	STIG	Configuration standards for DoD **IA** and IA-enabled devices/ systems. (Source: http://iase.disa.mil/stigs/index.html)
Serif Font		A serif is a feature of the letters in a given typeset. They appear at the end of lines within the letters. An example would be the letter T in Times New Roman - at the end of each horizontal line is a tick that hangs down (that is the serif). Serif fonts include ***Times New Roman***, ***Bookman Oldstyle***, and ***Courier***.  I1165
Server		A computer software application that carries out some task (i.e., provides a service) on behalf of yet another piece of software called a **client**.
Service		A service is an autonomous encapsulation of some business or mission functionality. The service concept includes the notion of service providers and service consumers interacting via well-defined reusable interfaces. ***Note:*** *See P1304:* ***Service-Oriented Architecture*** *in Part 1 for additional information concerning services including implementation characteristics.*
Service Access Point	SAP	SAP provides all of the information necessary for a user to access and consume a service. Includes the logical and physical location of the service on the net.
Service Definition Framework	SDF	SDF provides service users, customers, developers, providers, and managers with a common frame of reference. Its structure and methodology enable you to fully define the **Service Access Points** (SAPs) for the service.
Service Discovery	SD	Provides a ***yellow pages***, categorized by **DoD** function, enabling users to advertise and locate capabilities available on the network.
Service Level Agreement	SLA	A contractual vehicle between a service provider and a service consumer. It specifies performance requirements, measures of effectiveness, reporting, cost, and recourse. It usually defines repair turnaround times for users.

Service-Oriented Architecture	SOA	NESI describes SOA as an architectural style used to design, develop, and deploy information technology (IT) systems based on decomposing functionality into services with well-defined interfaces. ***Note:*** *See P1304:* ***Service-Oriented Architecture*** *in Part 1 for additional information.*
Service Registry		Provides descriptive information about a service, enabling the lookup and discovery of services.
Servlet		A Java program that extends the functionality of a Web server, generating dynamic content and interacting with Web applications using a request-response paradigm. (Source:http://java.sun.com/j2ee/1.4/docs/glossary.html)
Session		An interaction between system entities of finite duration, often involving a user, typified by the maintenance of some state of the interaction for the duration of the interaction. (Source:http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Simple Mail Transfer Protocol	SMTP	
Situation Awareness Data Link	SADL	An Enhanced Position Location and Reporting System (EPLRS) radio modified for use in an aircraft. SADL and EPLRS radios are used to establish a common secure tactical data link network. (Source: http://aatc.aztucs.ang.af.mil/aatcinfo.htm)
SOAP		SOAP Version 1.2 is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics. (Source: SOAP Version 1.2 Second Edition, http://www.w3.org/TR/soap12-part1/#intro) ***Note:*** *The World Wide Web Consortium (W3C) changed the name of this protocol from* ***Simple Object Access Protocol 1.1 (SOAP)*** *to* ***SOAP Version 1.2*** *in the current version.*
Software Communications Architecture	SCA	An implementation-independent framework for the development of software for an established hardware platform, such as software defined radios.

Software Component		A software component is a software system element offering a predefined service and able to communicate with other components. It is a unit of independent deployment and versioning, encapsulated, multiple-use, non-context-specific and composeable with other components. Source: http://en.wikipedia.org/wiki/Software_component#Software_component
Spyware		Any software that covertly gathers user information through the user's **Internet** connection without the user's knowledge, usually for advertising purposes. (Source: http://www.webopedia.com/TERM/s/spyware.html)
Stakeholder		An enterprise, organization, or individual having an interest or a stake in the outcome of the engineering of a system. (Source: EIA-632, Annex A)
Stored Procedure		A unit or module of code that executes in a database and implement some bit of application logic or business rule. Often written in proprietary language such as Oracle's PL/SQL or Sybase's Transact-SQL.
Structured Query Language	SQL	The standardized relational database language for defining database objects and manipulating data. (Source:http://java.sun.com/j2ee/1.4/docs/glossary.html)

Style Sheet		Style sheets describe how documents are presented on screens, in print, or perhaps how they are pronounced. (Source: http://www.w3.org/Style)
Surrogate Key		A surrogate key is a primary key that has been explicitly created and has no relationship with the naturally occurring data found within a table.



Surrogate Keys

Students:

Stu. ID	Name	Address	Phone
4321	John Public	200 Ash St, Hometown, USA	800-555-1234
1234	Jane Doe	170 Elm Ave, Hometown, USA	800-555-1212

Courses:

Stu. ID	Course #	Name
1234	B100	Intro Bio
1234	C100	Intro Chem
1234	P100	Intro Phy
1234	E100	English 1
4321	C100	Intro Chem
4321	P100	Intro Phy

If the student name "Jane Doe" changes, only one occurrence of the name must be changed.

I1167

See **Natural Key** and **Primary Key**.

Sustainment		One of the two major efforts (with disposal) of the Operations and Support phase of a DoD acquisition program. Sustainment includes supply, maintenance, transportation, sustaining engineering, data management, configuration management, manpower, personnel, training, habitability, survivability, environment, safety (including explosives safety), occupational health, protection of critical program information, anti-tamper provisions, and **Information Technology** (IT), including **National Security Systems** (NSS), supportability and interoperability functions. (Source: DoD Instruction 5000.2, 12 May 2003, *Operation of the Defense Acquisition System*, Section 3.9.2)
Symmetric Key Algorithm		Encryption algorithm where the same key is used for both encrypting and decrypting a message.
System		Two or more interrelated pieces of equipment (or sets) arranged in a package to perform an operational function or to satisfy a requirement. (Source: *Defense Acquisition Glossary of Terms*, Jan 2001)
System Component		A basic part of a system. System components may be personnel, hardware, software, facilities, data, material, services, and/or techniques that satisfy one or more requirements in the lowest levels of the functional

		architecture. System components may be subsystems and/or configuration items. ***Note:*** *See* ***component****.*
Systems and Services View	SV	The SV is a set of graphical and textual products that describes systems and interconnections providing for, or supporting, DoD functions. DoD functions include both warfighting and business functions. The SV associates systems resources to the Operational View (OV). These systems resources support the operational activities and facilitate the exchange of information among operational nodes. (Source: DoDAF v1.5 Volume I: Definitions and Guidelines, 23 April 2007)
Taxonomy		The science of categorization, or classification, of things based on a predetermined system. In reference to Web sites and portals, a site's taxonomy is the way it organizes its data into categories and subcategories, sometimes displayed in a site map. (Source: http://www.webopedia.com/TERM/t/taxonomy.html)
Taxonomy Gallery		The Taxonomy Gallery [of the **DoD Metadata Registry and Clearinghouse**] provides XML-based **taxonomy** files that describe one or more nodes in a hierarchical classification of items, and their relationships to other nodes. The taxonomy files registered with the Taxonomy Gallery are organized by governance namespace. (Source: http://www.disa.mil/nces/development/developer_doc_overview.html)
Technical Standards View	TV	The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed. The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture. (Source: *DoDAF* v1.5 Volume 1: Definitions and Guidelines, 23 April 2007)

Tenet		Net-centric design precept.
Topic		Topics are used to manage content flow between publishers and subscribers. Topics must be known in such a way that subscribers can refer to them unambiguously. In **DDS**, Topics conceptually fits between **publications** and **subscriptions** and associate a name (unique in the **domain**), a data-type, and **QoS** parameters related to the data.
Transaction		A set of input data that triggers execution of a specific processor job. Usually manipulates data that may need to be rolled back to the original values if any part of the transaction fails. Transactions enable multiple users to access the same data concurrently. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Transmission Control Protocol	TCP	One of the core protocols of the Internet protocol suite. Using TCP, programs on networked computers can create connections to one another, over which they can send data. The protocol guarantees that data sent by one endpoint will be received in the same order by the other, without any pieces missing. It also distinguishes data for different applications (such as a Web server and an email server) on the same computer. (Source: http://en.wikipedia.org/wiki/Transmission_Control_Protocol)
Transmission Control Protocol/Internet Protocol	TCP/IP	A suite of communications protocols used to connect hosts on the Internet. TCP/IP uses several protocols, the two main ones being TCP and IP. TCP/IP is built into the UNIX operating system and is used by the Internet, making it the de facto standard for transmitting data over networks. Even network operating systems that have their own protocols, such as Netware, also support TCP/IP.
Transport Layer Security	TLS	A protocol that guarantees privacy and data integrity between client/server applications communicating over the Internet. The TLS protocol is made up of two layers: • The TLS Record Protocol -- layered on top of a reliable transport protocol, such as TCP, it ensures that the connection is private by using symmetric data encryption and it ensures that the connection is reliable. The TLS Record Protocol also is used for encapsulation of higher-level protocols, such as the TLS Handshake Protocol. • The TLS Handshake Protocol -- allows authentication between the server and client and the negotiation of an encryption algorithm and cryptographic keys before the application protocol transmits or receives any data. (Source: http://www.webopedia.com/TERM/T/TLS.html)

Trigger		In a DBMS, a trigger is a SQL procedure that initiates (fires) an action when an event (INSERT, DELETE, or UPDATE) occurs. Since triggers are event-driven specialized procedures, the DBMS stores and manages them. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers maintain the referential integrity of data by changing the data in a systematic fashion.
Triple Data Encryption Algorithm	TDEA	An encryption algorithm whose key consists of three DES (Data Encryption Standard) keys, which is also referred to as a key bundle. A DES key consists of 64 binary digits ("0"s or "1"s) of which 56 bits are randomly generated and used directly by the algorithm. (The other 8 bits, which are not used by the algorithm, may be used for error detection.) Each TDEA encryption/decryption operation (as specified in ANSI X9.52) is a compound operation of DES encryption and decryption operations. Let EK(I) and DK(I) represent the DES encryption and decryption of ***I*** using DES key ***K*** respectively. (Source: http://www.atis.org/tg2k/_triple_data_encryption_algorithm.html)
Trusted Guard		Accredited to pass information between two networks at different security levels according to well defined rules and other controls. Guard products only pass defined types of information (e.g., email, images, or formatted messages). A key challenge is how to implement net-centric operations across trusted guards in the presence of **CES** services.
Trusted Path		A communications path where (1) there is reasonable confidence that there has not been any malicious alteration of the information; (2) the data are timely, meaning they originated within a small preceding period of time.
Trust Point		A trust point is a **Certificate Authority** (**CA**) that is the root of all trust for all CAs in a CA hierarchy.
Unclassified but Sensitive Internet Protocol Router Network	NIPRNet	NIPRNet provides seamless interoperability for unclassified combat support applications, as well as controlled access to the Internet. Direct connection data rates range from 56Kbps to 622Mbps. Remote dial-up services are available up to 56Kbps. (Source: http://www.disa.mil/main/prodsol/data.html)
Unified Modeling Language	UML	In the field of software engineering, the Unified Modeling Language (UML) is a standardized specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model. UML is officially defined at the **Object Management Group** (OMG) by the UML metamodel, a Meta-Object Facility metamodel (MOF). (Source: http://en.wikipedia.org/wiki/Unified_Modeling_Language; 30 March 2007)
Uniform Resource Locator	URL	A sequence of characters that represents information resources on a computer or in a network such as the Internet. This sequence of characters includes (1) the abbreviated name of the protocol used to access the information resource and (2) the information used by the

		protocol to locate the information resource.(Source: http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html)
UNIQUE Key Integrity Constraint		A **`UNIQUE`** key integrity constraint requires that every value in a column or set of columns (key) be unique; that is, no two rows of a table have duplicate values in a specified column or set of columns. (Source: http://www.lc.leidenuniv.nl/awcourse/oracle/server.920/a96524/c22integ.htm)
Universal Description, Discovery, and Integration	UDDI	An industry initiative to create a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet, as well as a registry. It is being developed by a vendor consortium. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
User Datagram Protocol	UDP	A connectionless protocol that, like **TCP**, runs on top of **Internet Protocol** (IP) networks. Unlike **Transmission Control Protocol/Internet Protocol** (TCP/IP), UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. It's used primarily for broadcasting messages over a network. (Source: http://www.webopedia.com/TERM/U/User_Datagram_Protocol.html)
Valid		A valid XML document has data that conforms to a particular set of user-defined content rules, or XML Schemas, that describe correct data values and locations. For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and it instead has the text ***hello***, is empty, or has other elements in its content, then the document is not valid. (Source: adapted from http://en.wikipedia.org/wiki/XML; 9/11/2006)
VBScript		A programming language developed by Microsoft that is similar to **JavaScript**. It is used to embed code into **HTML** pages. It is actually a subset of Microsoft's Visual Basic.
Vendor		Any person, organization, or automated asset that interfaces with the information environment as a service consumer or service provider.
Virtual Private Network	VPN	A network that is constructed by using public wires to connect nodes. For example, there are a number of systems that enable the creation of networks using the Internet as the medium for transporting data. These systems use encryption and other security mechanisms to ensure that only authorized users can access the network and that the data cannot be intercepted. (Source: http://www.webopedia.com/TERM/V/VPN.html)

Web Application		A collection of components that can be bundled together and run in multiple containers from multiple vendors. -OR- An application written for the Internet, including those built with Java technologies such as Java Server Pages and servlets, and those built with non-Java technologies such as CGI and Perl. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Web Application Archive	WAR	A **JAR** archive that contains a **Web module**. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)

Web Browser		A client program that initiates requests to a **Web server** and displays the information that the server returns. (Source: http://publib.boulder.ibm.com/infocenter/adiehelp/index.jsp?topic=/com.ibm.wsinted.glossary.doc/topics/glossary.html)
Web Container		A container that implements the Web-component contract of the **J2EE** architecture. This contract specifies a runtime environment for Web components that includes security, concurrency, life-cycle management, transaction, deployment, and other services. A Web container provides the same services as a **JSP** container as well as a federated view of the J2EE platform **APIs**. A Web container is provided by a Web or J2EE server. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Web Module		A deployable unit that consists of one or more Web components, other resources, and a Web application deployment descriptor. The Web module is contained in a hierarchy of directories and files in a standard Web application format. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Web Page		A document created with **HTML** (HyperText Markup Language) that is part of a group of hypertext documents or resources available on the World Wide Web. Collectively, these documents and resources form what is known as a **Web site**. You can read HTML documents that reside somewhere on the Internet or on your local hard drive with software called a **Web browser**. Web pages can contain hypertext links to other places within the same document, to other documents at the same Web site, or to documents at other Web sites.
Web Server		Software that provides services to access the Internet, an intranet, or an extranet. A Web server hosts **Web sites**, provides support for HTTP and other protocols, and executes server-side programs (such as **CGI** scripts or servlets) that perform certain functions. In the **J2EE** architecture, a Web server provides services to a **Web container**. For example, a Web container typically relies on a Web server to provide **HTTP** message handling. The J2EE architecture assumes that a Web container is hosted by a Web server from the same vendor, so it does not specify the contract between these two entities. A Web server can host one or more Web containers. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
Web Service		A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically **WSDL**). Other systems interact with the Web service in a manner prescribed by its description using **SOAP** messages, typically conveyed using **HTTP** with an **XML** serialization in conjunction with other Web-related standards. (Source: http://www.w3.org/TR/ws-gloss/)
Web Services Description Language	WSDL	WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either

		document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. (Source: W3C Note on WSDL 1.1 of 15 March 2001 http://www.w3.org/TR/wsdl)
Web Services for Interactive Applications	WSIA	
Web Services for Remote Portlets	WSRP	The WSRP specification defines a **Web service** interface for interacting with interactive presentation-oriented Web services. It has been produced through the joint efforts of the Web Services for Interactive Applications (**WSIA**) and Web Services for Remote Portals (WSRP) OASIS Technical Committees. Scenarios that motivate WSRP/ WSIA functionality include (1) **portal** servers providing **portlets** as presentation-oriented Web services that can be used by aggregation engines; (2) portal servers consuming presentation-oriented Web services provided by portal or non-portal content providers and integrating them into a portal framework. (Source: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf)
Web Services Interoperability Organization	WS-I	WS-I is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web services leaders helps customers to develop interoperable Web services by providing guidance, recommended practices and supporting resources. (Source: http://www.ws-i.org/about/Default.aspx)
Web Site		A Web site, website, or WWW site (often shortened to just "site") is a collection of Web pages (i.e., HTML/XHTML documents accessible via **HTTP** on the Internet). All publicly accessible Web sites in existence comprise the World Wide Web. The pages of a Web site are accessed from a common root URL, the homepage, and usually reside on the same physical server. The URLs of the pages organize them into a hierarchy, although the hyperlinks between them control how the reader perceives the overall structure and how the traffic flows between the different parts of the site. (Source: http://en.wikipedia.org/wiki/web_site)
Wireless Application Protocol	WAP	WAP is an open international standard for applications that use wireless communication, such as Internet access from a mobile phone. WAP provides services equivalent to a web browser with some mobile-specific additions. It is specifically designed to address the limitations of very small portable devices. During its first years of existence WAP suffered from considerable negative media attention and has been criticised heavily for its design choices and limitations. (Source: http://en.wikipedia.org/wiki/WAP)
Wireless Markup Language	WML	WML is the primary content format for devices that implement the **WAP** (Wireless Application Protocol) specification based on XML, such as mobile phones. (Source: http://en.wikipedia.org/wiki/Wireless_Markup_Language)

World Wide Web Consortium	W3C	The World Wide Web Consortium (W3C) is an international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards. W3C's mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web. (Source: http://www.w3.org/Consortium/)
XML Attribute		An XML structural construct. A name-value pair, separated by an equals sign, included inside a tagged element that modifies certain features of the element. All attribute values, including things like size and width, are in fact text strings and not numbers. For XML, all values must be enclosed in quotation marks. Attributes can be declared for an XML element type using an attribute list declaration. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)
XML Element		An XML structural construct. An XML element consists of a start tag, an end tag, and the information between the tags, which is often referred to as the contents. Each element has a type, identified by name, sometimes called its "generic identifier" (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value. An instance of an element is declared using <element> tags. Elements used in an XML file are described by a **DTD** or schema, either of which can provide a description of the structure of the data. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)
XML Gallery		The XML Gallery [of the **DoD Metadata Registry and Clearinghouse**] contains information resources such as submission packages, elements, attributes, and schemas that have been registered by DOD software developers. These information resources use XML, a platform and vendor independent format for exchanging data, to handle data, data structures, and data descriptions (metadata). (Source: http://www.disa.mil/nces/development/developer_doc_overview.html)
XML Information Resources		Document Type Definition (**DTD**) or XML Schema Documents (**XSD**) files.
XML Schema		A database-inspired method for specifying constraints on documents using an XML-based language. Schemas address deficiencies in **DTDs**, such as the inability to constrain the kinds of data that can occur in a particular field. Because schemas are founded on XML, they are hierarchical. Thus it is easier to create an unambiguous specification, and it is possible to determine the scope over which a comment is meant to apply. (Source: http://java.sun.com/j2ee/1.4/docs/glossary.html)
XML Schema Definition	XSD	A language proposed by the **W3C** XML Schema Working Group for use in defining schemas. Schemas are useful for enforcing structure and/or constraining the types of data that can be used validly within other XML documents. XML Schema Definition refers to the fully specified and currently recommended standard for use in authoring XML schemas. Because the XSD specification was only recently finalized,

		support for it was only made available with the release of MSXML 4.0. It carries out the same basic tasks as DTD, but with more power and flexibility. Unlike DTD, which requires its own language and syntax, XSD uses XML syntax for its language. XSD closely resembles and extends the capabilities of XDR. Unlike XDR, which was implemented and made available by Microsoft in MSXML 2.0 and later releases, the W3C now recommends the use of XSD as a standard for defining XML schemas. (Source: http://msdn2.microsoft.com/en-us/library/ms256452.aspx)

References

R1008	Web Services Security Specification, March 2004, (http://www.oasis-open.org/specs/index.php)
R1051	DoD Meta Data Registry for XSLT samples. [http://diides.ncr.disa.mil/mdregHomePage/mdregHome.portal]
R1070	**C2IEDM data model** specifications: http://www.mip-site.org/
R1131	XML Schema Part 2: Datatypes Second Edition - http://www.w3.org/TR/xmlschema-2/#built-in-datatypes
R1149	"Common Presentation Layer Guide Standard 03-01," NAVSEA, September 2006
R1155	"Electronic and Information Technology Accessibility Standards," Federal Register, [http://www.access-board.gov/sec508/508standards.pdf]
R1164	DoD Directive 5000.1, *The Defense Acquisition System*, 12 May 2003 (certified current as of 24 November 2003); http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf.
R1165	DoD Instruction 5000.2, *Operation of the Defense Acquisition System*, 12 May 2003; http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf.
R1166	DoD Directive 8100.1, *Global Information Grid (GIG) Overarching Policy*, 19 September 2002 (certified current as of 21 November 2003); http://www.dtic.mil/whs/directives/corres/pdf/810001p.pdf.
R1168	DoD Instruction 4630.8, *Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS)*, 30 June 2004; http://www.dtic.mil/whs/directives/corres/pdf/463008p.pdf.
R1171	*DoD Architecture Framework (DoDAF)*, Version 1.5, 23 April 2007; https://dars1.army.mil/IER/index.jsp
R1172	*DoD Net-Centric Data Strategy*, DoD Chief Information Officer, 9 May 2003, http://www.defenselink.mil/cio-nii/docs/Net-Centric-Data-Strategy-2003-05-092.pdf.
R1173	CJCSI 3170.01F, *Joint Capabilities Integration and Development System*, 01 May 2007; http://www.dtic.mil/cjcs_directives/cdata/unlimit/3170_01new.pdf.
R1174	CJCSM 3170.01C, *Operation of the Joint Capabilities Integration and Development System*, 01 May 2007; http://www.dtic.mil/cjcs_directives/cdata/unlimit/m317001.pdf.
R1176	*Net-Centric Operations and Warfare Reference Model (NCOW RM)*, v1.1, 17 November 2005.
R1177	*Net-Centric Checklist*, V2.1.3, Office of the Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer, 12 May 2004; http://www.defenselink.mil/cio-nii/docs/NetCentric_Checklist_v2-1-3_.pdf.
R1178	*A Modular Open Systems Approach (MOSA) to Acquisition*, Version 2.0, September 2004; http://www.acq.osd.mil/osjtf/mosapart.html.

R1179	*DoD IT Standards Registry* (*DISR*); http://disronline.disa.mil.
R1182	Office of the Under Secretary of Defense (USD) for Acquisition, Technology and Logistics (AT&L) memorandum, *Instructions for Modular Open Systems Approach (MOSA) Implementation*, 7 July 2004, available at www.acq.osd.mil/osjtf
R1184	Program Executive Office, Integrated Warfare Systems (PEO-IWS 7), *Naval Open Architecture Contract Guidebook*, Version 1.1, 25 October 2007 available via the Defense Acquisition University Acquisition Community Connection Web site (https://acc.dau.mil/oa) by following the "Policy & Guidance" link.
R1185	GAO Report to Congressional Committees, Weapons Acquisition, *DOD Should Strengthen Polices for Assessing Technical Data Needs to Support Weapon Systems*

R1199	DoD Instruction 8580.1, *Information Assurance (IA) in the Defense Acquisition System* ***This instruction implements policy, assigns responsibilities, and prescribes procedures necessary to integrate Information Assurance (IA) into the Defense Acquisition System; describes required and recommended levels of IA activities relative to the acquisition of systems and services; describes the essential elements of an Acquisition IA Strategy, its applicability, and prescribes an Acquisition IA Strategy submission and review process.***
R1202	**OMG** Data Distribution Service for Real-time Systems Version 1.2
R1204	24 June 2005, *Air Force Internet Protocol Version 6 (IPv6) Policy and Transition Plan Tasking*
R1205	June 2006, *DoD IPv6 Transition Plan*, Version 2.0
R1206	DoD Instruction 8520.2; 1 April 2004; *Public Key Infrastructure (PKI) and Public Key (PK) Enabling*; http://www.dtic.mil/whs/directives/corres/pdf/852002p.pdf
R1207	David Sprott *"Service Oriented Architecture: An Introduction for Managers"*; July 2004, http://www.ibm.com/services/us/bcs/pdf/soa-cbdi-report-2004-july.pdf
R1211	AFEI NCOIF report *"Industry Best Practices in Achieving Service Oriented Architecture (SOA)"*; April 22, 2005; http://www.afei.org/news/documents/IndustryBestPracticesforAchievingSOA_000.pdf
R1215	Yefim V. Natis, Gartner; *"Service-Oriented Architecture Scenario"*; 16 April 2003; http://www.gartner.com/DisplayDocument?doc_cd=114358
R1217	DoD 8320.02-G, April 12, 2006, ***Guidance for Implementing Net-Centric Data Sharing***; http://www.dtic.mil/whs/directives/corres/pdf/832002g.pdf
R1218	AF ESC Net-Centric Data Strategy Implementation Roadmap, Chief Architect's Office, 5/23/2003, Draft v 0.83
R1219	Dr. Mark Kramer, *"Towards Implementation of the DoD Net-Centric Data Strategy (NCDS)"*; May 2007 (presentation)
R1224	DoD Chief Information Officer Memorandum, ***DoD Net-Centric Data Management Strategy - Metadata Registration***, 3 April 2003 (available at http://www.defenselink.mil/cio-nii/docs/DMmemo20030403.pdf)
R1225	DoD Discovery Metadata Specification (DDMS); refer to the DDMS homepage for current version information: http://metadata.dod.mil/mdr/irs/DDMS/
R1226	XML Schema Best Practices - http://www.xfront.org
R1227	DoD Metadata Registry and Clearinghouse, http://xml.dod.mil
R1228	ISO/IEC Standard 11179, ***Information Technology # Metadata Registries (MDR)***, Parts 1-6, available from the International Organization for Standardization (ISO), http://www.iso.org
R1229	XML Schema Specification, World Wide Web Consortium (W3C), http://www.w3c.org/XML
R1232	DoD Directive 5230.9, ***Clearance of DoD Information for Public Release***, 09 April 1996
R1235	CJCSM 3170.01B, Operation of the Joint Capabilities Integration and Development System, 11 May 2005

R1237	Web Services Interoperability (WS-I) Basic Security Profile, http://www.ws-i-org
R1239	NCIDs Global Information Grid Net-Centric Iimplementation Document - Service Definition Framework (S300), 21 December 2005
R1240	ASD(NII)/DoDCIO Memo, Subject: Radio Frequency (RF) Equipment Acquisition Policy (JTRS), 17 June 2003
R1241	http://jtrs.army.mil (SCA)
R1243	Web Services Security (WSS) SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard 200401, March 2004 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0)
R1244	DISA ***Information Assurance Support Environment*** Web site, http://iase.disa.mil
R1245	DoD Directive 8320.2, ***Data Sharing in a Net-Centric Department of Defense***, December 2, 2004
R1246	SAML Token Profile, Working Draft 15, 19 July 2004 (Web Services Security); http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-1.0
R1247	Director of Central Intelligence Directive 6/3, ***Protecting Sensitive Compartmented Information within Information Systems***, 5 June 1999
R1249	Air Force Instruction 33-202, "Network and Computing Security", 26 September 2003
R1251	DoD CIO Guidance and Policy Memorandum 6-8510, "DoD GIG Information Assurance", 16 June 2000.
R1252	DoD End-to_End Information Assurance of GIG, Version 1.0, 30 June 2004
R1254	The Department of Defense (DoD) Internet Protocol Version 6 (IPv6) Transition Plan, March 2005
R1255	Green, David and Grillo, Bob, SRI International, "The State of IPv6 - A DoD Prospective, February 2005 (prepared for the DoD IPv6 Standards Working Group
R1256	International Organization for Standardization (ISO) Open Systems Interconnection Basic Reference Model (OSI Model)
R1257	Blake, S., Black D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, ***An Architecture for Differentiated Services***, RFC 2475, IETF, December 1998.
R1262	The TeleManagement Forum's Enhanced Telecom Operations MapTM (eTOM) and the Information Technology Infrastructure Library (ITILR)
R1283	Net-Centric Environment Joint Functional Concept, Version 1.0, April 7, 2005
R1284	Net-Centric Operational Environment Joint Integrating Concept, Version .08, August 26, 2005
R1288	Deputy Under Secretary of Defense for Advanced Systems and Concepts, ***Open Technology Development Roadmap Plan***, April 2006; http://www.acq.osd.mil/jctd/articles/OTDRoadmapFinal.pdf

R1291	DoD Instruction 8510.01, DoD Information Assurance Certification and Accreditation Process (DIACAP), 28 November 2007; available at http://www.dtic.mil/whs/directives/corres/pdf/851001p.pdf (superseded DoD Instruction 5200.40, DITSCAP)
R1307	IBM, ***Open Architecture Principles and Guidelines***, v1.5.4, 19 September 2007; available at http://www.acq.osd.mil/jctd/articles/OTDRoadmapFinal.pdf